Security Now! #985 - 07-30-24

Platform Key Disclosure

This week on Security Now!

The obligatory follow-up on the massive CrowdStrike event: How do CrowdStrike's users feel? Are they switching or staying? How does CrowdStrike explain what happened? and does it make sense? How much blame should they receive? An update on how Entrust will be attempting to keep its customers from changing certificate authorities. Firefox appears not to be blocking 3rd-party tracking cookies when it claims to be. How hiring remote workers can come back to bite you in the you-know-what. Did Google really want to kill off 3rd-party cookies or are they actually happy? And is there any hope of ending abusive tracking? Auto-updating anything is fraught with danger. Why do we do it and is there no better solution? And what serious mistake did a security firm discover that compromises the security of nearly 850 PC makes and models?

That’s right... because Large Print is what one looks for in an Audio Book.

Security News

This is the note that last week I mentioned I would be sharing this week. It was DM'ed to me on Twitter, and I do not, never have and apparently never will inhabit Twitter, I didn't discover it in time to include in last week's podcast. Here's what our listener had to say about CrowdStrike:

Hi Steve! I am writing to you from New South Wales, Australia. I don't really use twitter… but here I am sending you a DM. Without a doubt you will be mentioning the Crowdstrike Outage in your Security Now this week. I thought to give you an industry perspective here.

I work in Cybersecurity at a large enterprise organization with the Crowdstrike Falcon agent deployed across the environment. Think approximately 20,000 endpoints. Around 2-3pm Sydney the BSOD wave hit Australia. The impact cannot be overstated. All windows systems in our network started BSOD-ing at once. This is beyond a horrible outage. Crowdstrike will need to be held accountable and show they can improve their software stability (rather than sprucing AI mumbo jumbo) if they want to remain a preferred provider.

Okay. In defense of Crowdstrike, their Falcon EDR tool is nothing short of amazing. The monitoring features of Falcon are at the top. It monitors file creation, network processes, registry edits, DNS requests, process executions, scripts, https requests, logons, logoffs, failed logons and so much more. The agent also allows for containing infected hosts, blocking indicators of compromise, opening a command shell on the host, collecting forensic data, submitting files to a malware sandbox, automation workflows, an API, powershell/python/go libraries, it integrates with threat intel feeds and more and more.

Most importantly: Crowdstrike has excellent customer support. Their team is helpful, knowledgeable and responsive to questions or feature requests. Despite this distavorous outage, we are a more secure company for using Crowdstrike. They have saved our butts numerous times. And I am sure other enterprises feel the same. (Full disclosure: I do not work for Crowdstrike but I do have one of their T-shirts.)

Why am I telling you this?

Because the second in line competitor to Crowdstrike Falcon is Microsoft Defender for Endpoint (MDE). MDE is not even close to offering Crowdstrike Falcon's level of protection.

Even worse, Microsoft's customer support is atrocious to the point of being absurd. I have interacted with Microsoft's security support numerous times. They were unable to answer even the most basic questions about how their own product operated and often placed me in an endless loop of escalating my problem to another support staff, forcing me to re-explain my problem to the point where I gave up asking for their help. I even caught one of their support users using ChatGPT to respond to my emails. And this is with an enterprise-level support plan.

As the dust starts to settle after this event, I imagine Microsoft will begin a campaign of aggressively selling Defender for Endpoint. Falcon is often more expensive than MDE, since Microsoft provides significant discounts depending on the other Microsoft services a customer consumes. Sadly, I imagine many Executive leadership teams will be dumping Crowdstrike after this outage and signing on with MDE. Relying on Microsoft for endpoint security will further inflate the single point of failure balloon that is Microsoft – leaving us all less secure in the long run.

Finally, I'm a big fan of the show. I started listening around 2015. As a result of listening to your show, I switched to a career in Cybersecurity. Thank you Leo and Steve 😊

This all rings 100% true to me. This is the correct way for a company like CrowdStrike to survive and to thrive. They really are offering dramatically more value and functionality than Microsoft. So the income they are earning is actually earned and deserved.

One key bit of information we're missing is whether all of these Windows systems, servers and networks that are constantly being overrun with viruses and ransomware and costing their enterprises tens of millions of dollars to restore and recover, are protected with CrowdStrike. If CrowdStrike is actually successfully blocking enterprise-wide catastrophe hour by hour and day by day, then that significantly factors into the risk/reward calculation. Our CrowdStrike user wrote: "*They have saved our butts numerous times. And I am sure other enterprises feel the same.*" THAT is a crucially important fact that is easily missed.

It may well be that right now corporate CIOs are meeting with their other C-suite executives and boards and reminding them that while, yes, what happened 10 days ago was bad, but even so it's worth it because this same system had previously prevented, say, 14 separate known instances of internal and external network breach, any of which may have resulted in all servers and workstation being irreversibly encrypted, public humiliation and demands for illegal ransom payment to Russia. So if that hasn't happened because, as our listener wrote *"CrowdStrike saved our butts numerous times"* then the very rational decision might well be to stick with this proven solution in the knowledge that CrowdStrike will have definitely learned a very important and valuable lesson and will be arranging to never allow anything like this to ever happen again.

If it never happens again then remaining with this superior solution is the obvious choice. But if it should ever happen again then in retrospect remaining with them will have been difficult to justify and I would not be surprised if people were fired over their decision not to leave CrowdStrike after this first major incident. But even so, if CrowdStrike's customers are able to point to the very real benefits they have been receiving from the use of this somewhat dangerous system, then even so it might be worth remaining with it.

Before we talk about CrowdStrike's response I want to share another interesting and important piece of feedback from a listener and something I found on Reddit. Our listener Dan Moutal sent:

I work at a company that uses CrowdStrike and I thought you would appreciate some insight. Thankfully we were only minimally affected as many of our Windows users were at a team-building event with their laptops powered down and not working, and our servers primarily run linux. So only a handful of workstations were affected. However the recovery was hampered by the common knowledge (which turned out to be false) that the Bitlocker recovery key would be needed to boot into safe mode.

When you try to boot into safe mode in Windows you are asked for the Bitlocker recovery key. Most knowledgebase articles (even from Microsoft) state that you need to enter the Bitlocker key at this point. ***But this is not required,*** *it's just not-obvious and not well known how to bypass this.*

Here's what we discovered over the weekend:

Cycle through the blue-screen error (when the host continues to crash) until you get to the recovery screen. Perform the following steps: below:

1. *Navigate to Troubleshoot > Advanced Options > Startup Settings*
2. *Press: Restart*
3. *Skip the first Bitlocker recovery key prompt by pressing ESC.*
4. *Skip the second Bitlocker recovery key prompt by selecting "Skip This Drive" at the bottom right*
5. *Navigate to Troubleshoot > Advanced Options > Command Prompt*
6. *Type: bcdedit /set {default) safeboot minimal then press Enter.*
7. *Close the command prompt window by clicking the X in the top right. This will return you back to the blue screen (WinRE main menu)*
8. *Select: Continue.*

Your PC will now reboot; it may cycle 2-3 times. Your PC should now boot into safe mode.

I confirmed this worked, and allowed us to recover a few systems where we did not have the bitlocker keys available.

I think Microsoft deserves a lot of blame for the poor recovery process when safemode is needed. They should not be asking for Bitlocker keys if they are not needed. At the bare minimum they need to make this knowledge much more well known so system admins who don't have bitlocker keys can still boot into safemode when disaster strikes.

I also want to send a shout out to CrowdStrike's technical support team. I am sure they were swamped by support requests on Friday, but despite that we were talking to them after waiting on hold for only 10 minutes, and they were very helpful. Most vendors aren't that quick or usefull on a good day, let alone on a day when they are the cause of a massive outage. CrowdStrike is an expensive product, but it is clear that a large chunk of that expense pays for a large and well trained support staff.

And from a post on Reddit which I've edited a bit for the podcast:

Just exited a meeting with Crowdstrike. You can remediate all of your endpoints from the cloud. If you're thinking, "That's impossible. How?", this was also the first question I asked and they gave a reasonable answer.

To be effective, Crowdstrike services are loaded very early on in the boot process and they communicate directly with Crowdstrike. This communication is use to tell crowdstrike to quarantine windows\system32\drivers\crowdstrike\c-00000291.sys*

To do this, you must first opt-in by submitting a request via the support portal, providing your customer IDs and request to be included in cloud remediation.

At the time of the meeting, the average wait time for inclusion was less than one hour. Once you receive email indicating that you have been included, you simply have your users reboot their computers.

CloudStrike noted that sometimes the boot process completes too quickly for the client to get the update and a 2nd or 3rd try is needed, but it is working for nearly all of their users. At the time of the meeting, they had remediated more than 500,000 endpoints this way.

It was suggested to use a wired connection when possible since wifi connected users have the most frequent trouble, probably because WiFi connectivity becomes available later in the boot process and after a crash will have occurred.

This also works with home and remote users since all they need is an internet connection. The point is, they do not need to be and should not be VPN'd into the corporate network.

I thought that was interesting since we essentially have another of those "race conditions" we've talked about recently. In this case it's one where we're hoping that the CrowdStrike network-based update succeeds before the crash can occur.

Okay. So what more do we know today than we did one week ago at the time of last week's podcast?

The questions that were on everyone's mind were variations of "How could this possibly have been allowed to happen in the first place?" "How could CrowsStrike not have had measures in place to prevent this?" and "Even staggering the release of the buggy file would have limited the scale of the damage. Why wasn't that, at least, part of their standard operating procedure?"

For last week's podcast we had no answers to any of those questions. Among the several possibilities, I suggested that if they did have some pre-release testing system in place then it must have somehow failed. And that's the explanation that the industry has received. I have no doubt about its truth, since CrowdStrike executives will be repeating that under oath shortly.

Last week we shared what little was known, which CrowdStrike had published by that point under the title "What Happened?" and we also had the statement from George Kurtz, CrowdStrike's founder and CEO. This week, their "What Happened?" is followed by "What Went Wrong and Why?" Despite the fact that it contains a bunch of eye-crossing jargon which sounds like gobbledygook I think it's important for everyone to hear what CrowdStrike said. They wrote:

CrowdStrike delivers security content configuration updates to our sensors in two ways: Sensor Content that is shipped with our sensor directly, and Rapid Response Content that is designed to respond to the changing threat landscape at operational speed.

The issue on Friday involved a Rapid Response Content update with an undetected error.

Sensor Content provides a wide range of capabilities to assist in adversary response. It is always part of a sensor release and not dynamically updated from the cloud. Sensor Content includes on-sensor AI and machine learning models, and comprises code written expressly to deliver longer-term, reusable capabilities for CrowdStrike's threat detection engineers.

These capabilities include Template Types, which have pre-defined fields for threat detection engineers to leverage in Rapid Response Content. Template Types are expressed in code. All Sensor Content, including Template Types, go through an extensive QA process, which

includes automated testing, manual testing, validation and rollout steps.

The sensor release process begins with automated testing, both prior to and after merging into our code base. This includes unit testing, integration testing, performance testing and stress testing. This culminates in a staged sensor rollout process that starts with dogfooding internally at CrowdStrike, followed by early adopters. It is then made generally available to customers. Customers then have the option of selecting which parts of their fleet should install the latest sensor release ('N'), or one version older ('N-1') or two versions older ('N-2') through Sensor Update Policies.

The event of Friday, July 19, 2024 was not triggered by Sensor Content, which is only delivered with the release of an updated Falcon sensor. Customers have complete control over the deployment of the sensor — which includes Sensor Content and Template Types.

Rapid Response Content is used to perform a variety of behavioral pattern-matching operations on the sensor using a highly optimized engine. Rapid Response Content is a representation of fields and values, with associated filtering. This Rapid Response Content is stored in a proprietary binary file that contains configuration data. It is not code or a kernel driver.

Rapid Response Content is delivered as "Template Instances," which are instantiations of a given Template Type. Each Template Instance maps to specific behaviors for the sensor to observe, detect or prevent. Template Instances have a set of fields that can be configured to match the desired behavior.

In other words, Template Types represent a sensor capability that enables new telemetry and detection, and their runtime behavior is configured dynamically by the Template Instance (in other words, the Rapid Response Content).

Rapid Response Content provides visibility and detections on the sensor without requiring sensor code changes. This capability is used by threat detection engineers to gather telemetry, identify indicators of adversary behavior and perform detections and preventions. Rapid Response Content is behavioral heuristics, separate and distinct from CrowdStrike's on-sensor AI prevention and detection capabilities.

Rapid Response Content is delivered as content configuration updates to the Falcon sensor. There are three primary systems: the Content Configuration System, the Content Interpreter and the Sensor Detection Engine.

The Content Configuration System is part of the Falcon platform in the cloud, while the Content Interpreter and Sensor Detection Engine are components of the Falcon sensor. The Content Configuration System is used to create Template Instances, which are validated and deployed to the sensor through a mechanism called Channel Files. The sensor stores and updates its content configuration data through Channel Files, which are written to disk on the host.

The Content Interpreter on the sensor reads the Channel File and interprets the Rapid Response Content, enabling the Sensor Detection Engine to observe, detect or prevent malicious activity, depending on the customer's policy configuration. The Content Interpreter is designed to gracefully handle exceptions from potentially problematic content.

Newly released Template Types are stress tested across many aspects, such as resource utilization, system performance impact and event volume. For each Template Type, a specific Template Instance is used to stress test the Template Type by matching against any possible value of the associated data fields to identify adverse system interactions.

Template Instances are created and configured through the use of the Content Configuration System, which includes the Content Validator that performs validation checks on the content before it is published.

Then they lay out a timeline of events which I'll make a bit easier to understand.

Recall that there was mention of named pipes being involved with this trouble. And I explained that named pipes were a very common means for different independent processes to communicate with each other. So way back on February 28th of this year CrowdStrike sensor v7.11 was made generally available to customers. It introduced a new Inter Process Communication (IPC) Template Type which was designed to detect novel attack techniques that abuse Named Pipes. This release followed all Sensor Content testing procedures outlined above in the Sensor Content section.

A week later, on March 5th, 2024, a stress test of the IPC Template Type was executed in our staging environment, which consists of a variety of operating systems and workloads. The IPC Template Type passed the stress test and was validated for use.

That same day, on following the successful stress testing, the first of the IPC Template Instances were released to production as part of a content configuration update. After that, three additional IPC Template Instances were deployed between April 8, 2024 and April 24, 2024. These Template Instances performed as expected in production.

Then, on that fateful day of Friday July 19th, two additional IPC Template Instances were deployed. One of these two was malformed and should never have been released. But due to a bug in the Content Validator, that malformed IPC Template Instance erroneously passed validation... despite containing problematic content data.

Based on the testing performed before the initial deployment of the new IPC Template Type back on March 5th, and in the trust in the checks performed by the Content Validator, and the several previous successful IPC Template Instance deployments, on Friday the 19th, these two instances, one of them malformed, were released and deployed into production.

When received by the sensor and loaded into the Content Interpreter, problematic content in Channel File 291 resulted in an out-of-bounds memory read triggering an exception. This unexpected exception could not be gracefully handled, resulting in a Windows operating system crash (BSOD).

So of course they then write under the title: *"How Do We Prevent This From Happening Again?"* Under the sub-head of "Software Resiliency and Testing" they have bullet points:

- *Improve Rapid Response Content testing by using testing types such as:*
  - *Local developer testing*
  - *Content update and rollback testing*
  - *Stress testing, fuzzing and fault injection*

    - *Stability testing*
    - *Content interface testing*

- *Add additional validation checks to the Content Validator for Rapid Response Content. A new check is in process to guard against this type of problematic content from being deployed in the future.*
- *Enhance existing error handling in the Content Interpreter.*

And under the sub-head of "Rapid Response Content Deployment" they have four items:

- *Implement a staggered deployment strategy for Rapid Response Content in which updates are gradually deployed to larger portions of the sensor base, starting with a canary deployment.*
- *Improve monitoring for both sensor and system performance, collecting feedback during Rapid Response Content deployment to guide a phased rollout.*
- *Provide customers with greater control over the delivery of Rapid Response Content updates by allowing granular selection of when and where these updates are deployed.*
- *Provide content update details via release notes, which customers can subscribe to.*

The bottom line to all this is that CrowdStrike now promises to do what it should have been doing all along. Of course, this is easy to say in retrospect, but you can bet that they themselves are wishing more than anyone else that they, too, had been doing all of this all along.

So is this another of those small earthquake tremors I was recently talking about? I suppose that would depend very much upon whom you ask. The source of the problem was centralized but the remediation of the problem was widely distributed. Across 8.5 million machines, several hundred thousand individual technicians got to work figuring out what had happened and each repairing the machines over which they had responsibility. As we know in the aftermath, some users of Windows appear to have been more seriously damaged than others. In some cases machines that were rebooted repaired themselves by retrieving the updated and repaired Template File. Whereas in other situations, such as with Delta Airlines, the effects from having Windows system crash lasted many days.

I have no direct experience with CrowdStrike. But not a single one of our listeners from whom we have heard, even after enduring this pain, sounded like they would prefer to operate without CrowdStrike-level protection and monitoring in the future. And I think that's a rational position. No one was happy that this occurred, and it really is foreseeable that CrowdStrike has learned a valuable lesson about using belts, suspenders, Velcro and some epoxy. They may have grown a bit too comfortable over time... but I'll bet that's been shaken out of them today.

Another bit of feedback: Since it's relevant to the CrowdStrike discussion, I wanted to share what another listener of ours, Vernon Young, shared with me. Vernon wrote:

> *Dear Steve, I am the IT Director for a high school and manage 700 computers and 50 virtual servers. A few weeks before the Kaspersky ban was announced, I placed a $12,000 renewal order with Kaspersky--$12,000 which will now be lost since the software won't work after*

September. After the ban was announced, I started looking for alternatives. I decided on Thursday, July 18 to go with CrowdStrike, the day before the world collapsed. Thankfully, I didn't feel like walking to the copier to scan the purchase order to send to the sales rep before I left for the day. Needless to say, I changed my mind Friday morning.

I cannot imagine being Vernon and needing to corral a high school campus full of mischievous and precocious high schoolers who imagine they're more clever than the rest of the world and whose juvenile brains' sense of right and wrong hasn't yet had the chance to fully develop. But think about the world Vernon is facing. He invests $12,000 to renew the school's Kaspersky A/V system license, only to have that lost. Then decides that CrowdStrike looks like the best alternative, only to have it collapse the world.

For what it's worth, I stand by the way I ended that CrowdStrike discussion. I would go with them today. All of the feedback we've received suggests that they are top notch and they are very clearly raising the bar to prevent another mistake from ever slipping past. There's just no way that they haven't ***really*** learned a lesson from this debacle, and that's all anyone can ask at this point.

Marcus Hutchins on who is to blame (https://grc.sc/985)
Last night, Marcus Hutchins posted a wonderfully comprehensive 18-minute YouTube video which thoroughly examines, explores and explains the history of the still-raging three-way battle among Microsoft, 3rd-party antivirus vendors and malware creators. It is this week's GRC shortcut of the week, so your browser can be redirected to it if you go to GRC.SC/985. When you go there be prepared for nearly 18 minutes of high-speed non-stop perfectly articulated techie detail, because that's what you're going to get.

Summarizing Marcus' explanation, since the beginning of Windows, the appearance of viruses and other malware, and the emergence of a market for 3rd-party antivirus vendors, there has been an uncomfortable relationship between Microsoft and 3rd-party A/V. To truly get the job done correctly, 3rd-party A/V has always needed deeper access into Windows than Microsoft has been willing to give. And the CrowdStrike incident shows what can happen when a 3rd-party makes a mistake in the Windows kernel. But it is not true that 3rd-party antivirus vendors **can** do the same thing as Microsoft can without being in the kernel. This is why Microsoft themselves do not use the APIs they have made available to other antivirus vendors. Those APIs do not get the job done. And the EU did **not** say that Microsoft had to make the kernel available to other 3rd-parties. The EU merely said that Microsoft needed to create a level playing field where the same features would be available to 3rd-parties as they were using themselves. Since Microsoft was unwilling to use only their own watered-down antivirus APIs and needed access to their own OS kernel, that same EU-mandated access has remained available to 3rd-party vendors as well.

Any comprehensive retelling of the saga of Windows kernel access must recognize that this area of Windows has been constantly evolving. Marcus notes that Windows Vista was a real mess, that many changes have been made since then, and that the latest Windows 10 1703 has made some changes that might offer some hope of a more stable world in the future. The problem, of course, is that 3rd-parties will still need to be offering their solutions on older Windows platforms which are still running just fine, refuse to die, and may not be upgradeable to later versions.

Entrust's Updated Info
Held a 10am "Webinar" last week which included the description of their solution with the partnership of SSL.com. It was largely what I presumed from what they had said earlier, which was that behind the scenes the Certificate Authority SSL.com would be creating and signing the certificates that Entrust would be purchasing from SSL.com and effectively re-selling. There were, however, two additional details that were interesting.

Before any certificate authority can issue domain validation certificates, the applicant's control over the domain name in question must be demonstrated. And SSL.com is not willing to take Entrust's word for it. So the additional wrinkle that will exist for any Entrust customers who wish to purchase webserver certificates from Entrust after this coming October 31st is that they will need to prove their domain ownership not to Entrust, as they have in the past, but to SSL.com.

The second wrinkle is that Entrust does not want SSL.com's name to appear in the web browser when a user inspects the security of their connection and to see who issued a site's certificate. So, although SSL.com will be creating each entire certificate for Entrust, they've agreed to have SSL.com embed an Entrust intermediate certificate into the certificate chain, since web browsers only show the signer of the webserver's final certificate in the chain. By placing Entrust in the middle, SSL.com will be signing the Entrust intermediary and Entrust's intermediary will be signing the server's domain certificate. In this way, it will be Entrust's name that will be seen by anyone checking.

3rd-Party Cookie Surprise
Something is going on with Firefox that is not clear. After last week's discussion of 3rd-party cookies, and Leo's playing with GRC's Cookie Forensics pages, several people commented that Firefox did not appear to be doing the right thing when it came to blocking 3rd-party cookies in what it calls "Strict" mode.
Strick mode is what I want but, sure enough, Strict mode behavior does not appear to be what I'm getting.

Under Firefox's "Enhanced Tracking Protection" we have Standard, Strict and Custom modes. Standard is described as "Balanced for protection and performance. Pages will load normally." However, Strict is described as "Stronger protection, but may cause some sites to content to break." And then it details this further, saying: Firefox blocks the following:

- Social media trackers
- Cross-site cookies in all windows
- Tracking content in all windows
- Cryptominers
- Fingerprinters

That all sounds great. The problem is, it doesn't appear to be working at all under Firefox. This issue arose in GRC's old school newsgroups so it grabbed a lot of attention and many others have confirmed that Firefox's Strict mode is apparently not doing what we want or expect. Chrome and Bing work perfectly.

In order to get Firefox to actually block 3rd-party cookies it's necessary to switch to custom mode, tell it that you want to block cookies, then under which types of cookies to block you cannot choose *"Cross-site tracking cookies"* you need to turn up the strength higher. Choosing *"Cross-site tracking cookies, and isolate other cross-site cookies"* also does not work. Neither does the setting *"Cookies from unvisited websites"*. Nope. It's necessary to choose Custom mode and the cookie blocking selection: *"All cross-site cookies (may cause websites to break)"*.

Once that's done, GRC's Cookie Forensics page shows that NO 3rd-party session or persistent cookies are being returned from Firefox. They are blocked completely. Back when I first wrote this, when 3rd-party cookies were disabled, some of the broken browsers would stop accepting new cookies but they would continue sending any cookies they already had. Happily, those days are gone, but this is disturbing behavior from Firefox and it's much worse than Chrome and Bing which behave perfectly.

Looking at the wording, which specifically refers to cross-site ***tracking*** cookies, it appears that Firefox may be making some sort of value judgment about which 3rd-party cross-site cookies are being used for tracking and which are not. That seems like a bad idea. Do they imagine that they have and can maintain a comprehensive list of tracking domains and won't allow 3rd-party cookies to be set by any of those? That's the only thing that comes to mind. That seems dumb.

There may be more to what's going on here, though. One person in GRC's newsgroup said that they set up a new virtual machine, installed Firefox, and it IS working correctly. That suggests that we may have another of those situations we've encountered in the past where less secure behavior is allowed to endure in the interest of not breaking anything in an existing installation whereas anything new is run under the improved security settings. But that's intolerable, too, because it appears to be completely transparent and if that's really what's going on, Firefox's user interface is not telling the truth.

I wanted to bring this up and put it on everyone's radar in case others like me were trusting and believing Firefox's meaning of the word "Strict". I would imagine that some of our listeners will be interested enough to dig into this and see whether they can determine what's going on. As everyone knows, I now have an effective incoming channel for effortlessly sending and receiving email.

Security training firm mistakenly hires a North Korean attacker

PC Magazine brings us the story of a security training firm who inadvertently hired a remote software engineer only to later discover that he was an imposter based in North Korea:

> *A US security training company discovered it mistakenly hired a North Korean hacker to be a software engineer after the employee's newly issued computer became infected with malware.*
>
> *The incident occurred at KnowBe4, which develops security awareness programs to teach employees about phishing attacks and cyber threats. The company recently hired a remote software engineer who cleared the interview and background check process. But last week, KnowBe4 uncovered something odd after sending the employee a company-issued Mac.*

KnowBe4 wrote in a blog post last Tuesday: "The moment it was received, it immediately started to load malware."

The company detected the malware thanks to the Mac's onboard security software. An investigation, with the help of the FBI and Google's security arm Mandiant, then concluded that the hired software engineer was actually a North Korean posing as an IT worker.

Fortunately, the company remotely contained the Mac before the hacker could use the computer to compromise KnowBe4's internal systems. When the malware was first detected, the company's IT team initially reached out to the employee, who claimed "that he was following steps on his router guide to troubleshoot a speed issue." But in reality, KnowBe4 caught the hired worker manipulating session files and executing unauthorized software, including using a Raspberry Pi to load the malware.

In response, KnowBe4's security team tried to call the hired software engineer, but he "stated he was unavailable for a call and later became unresponsive."

KnowBe4 says it shipped the work computer "to an address that is basically an 'IT mule laptop farm,'" which the North Korean then accessed via VPN.

Although KnowBe4 managed to thwart the breach, the incident underscores how North Korean hackers are exploiting remote IT jobs to infiltrate US companies. In May, the US warned that one group of North Koreans had been using identities from over 60 real US citizens to help them snag remote jobs.

The remote jobs can help North Korea generate revenue for their illegal programs and provide a way for the country's hackers to steal confidential information and pave the way for other attacks. In the case of KnowBe4, the fake software engineer resorted to using an AI-edited photo of a stock image to help them clear the company's interview process.

This story should bring a chill to anyone who might ever hire someone sight unseen based upon information that's available online – as opposed to the old fashioned way of taking a face-to-face meeting to interview the person and discuss how and whether there might be a good fit. One of the things we know is going on more and more is domestic firms dropping their in-house teams of talent in favor of off-shoring their needs as a means of increasing their agility and reducing their fixed costs. This is one of those things that accountants think is a great idea and I suppose there may be some places where this could work. But remote software development? I'd sure be wary about that.

The new tidbit that really caught my attention was the idea of something that was described as *"an IT mule laptop farm"*. Whoa. So this is at a benign location, where the fake worker says they're located. Being able to receive a company laptop at that location further solidifies the online legend they've erected. So this laptop is received by confederates who set it up in the IT mule farm and install VPN software, or perhaps attach the laptop to a remote KVM-over-IP system to keep the laptop completely clean. Either way, this allows the fake worker to appear to be using the laptop from the expected location when, instead, they are half-a-world away in a room filled with North Korean hackers all working diligently to attack the West. I wish this was just a 'B'-rated Sci-Fi movie, but it's all for real and it's happening as we speak.

Closing The Loop

A listener named Robert wrote:

Hello Steve! I'll try to not take too much of your time, but I'd like to mention one thing that irked me about the entire "Google trying to eradicate 3rd party cookies is a good thing" business. TL;DR: Google tries to get rid of 3rd party cookies to gain a monopoly in the ad market, not to protect users.

[I don't see that at all... but okay.]

*First and foremost: eradicating 3rd party cookies is a good thing — as a vehicle to stop tracking of website visitors. The only reason why Google would be *able* to actually force website owners to move from their cookie-based ad strategies to something else (floc, labels, whatever they call it) is that they have a near-monopoly in the browser market.*

[Absolutely 100% agreed. The only way the world would ever drop 3rd-party cookies would be if it was forced to and at this time in history, only Google is in the position to have the market power to force such a large change.]

It's important to keep in mind that Google is still a company that makes most of their money selling ads.

[Right.]

Every move they had made so far smelled like they wanted to upgrade their browser monopoly also into a ad tech monopoly. My suspicion is that it wasn't necessarily the ad companies directly that threatened Google about its plan to eradicate 3rd party cookies, but rather some pending monopoly concern about the ad market. But maybe I'm just too optimistic about that 🙂 So, well, just a thought I felt was a bit underrepresented. Thanks again for all the work! Robert

The problem I have with Robert's analysis is that I cannot see how Google is giving itself any special privileges through the adoption of their Privacy Sandbox. While it's absolutely true that they were dramatically changing the rules, everything that they were doing was a 100% open process, with open design and open discussion and open source. And they, themselves, were also being forced to play by those same new rules that they were asking everyone else to play by. So there was no advantage that they had over any other advertiser just because it was their web browser. And had they been successful in bringing about this change, the other browsers would have eventually adopted the same open Privacy Sandbox technologies.

I did not have time to address this fully last week due to the CrowdStrike event, so I'm glad for Robert's note. The EU bureaucrat's apparent capitulation to the slimy tracking and secretive user-profiling underworld, which in turn forced Google's browser to retain its historically abused 3rd-party cookie support, represents a massive privacy loss to the world. This was the wrong outcome and I sincerely hope that it is only a setback that will not stand for long.

Lisa in Worcester, Massachusetts wrote:

Steve, Another intriguing podcast, many thanks! I find it interesting the influence Google has and doesn't have. It seems more powerful over one company like Entrust, than a whole market like third party cookies. Is it influence or is it calculated cost benefit/loss that helps

Google/Alphabet decide where to flex its muscles. Thoughts from Worcester, MA — Lisa

As an observer of human politics I often observe the simple exercise of power. In U.S. politics we see this all the time. Both major political parties scream at each other crying foul and unfair, but they each do what they do simply because they want to and they can when they have the power.

I suspect the same is true with Google. Google has more power than Entrust but less power than the European Union. So Google was able to do what it wished with Entrust whereas the EU had the power to force Google to do what it wished. And who knows what's really going on inside Google? I very much wanted to see the end of 3rd-party cookie abuse. But we don't **really** know that Google, or at least that all of Google did. Others are suspicious of Google's motives and maybe they're right to be. The EU's pushback against Google's Privacy Sandbox might not be such a bad thing for Google. I would imagine that an entity the size of Google has plenty of its own internal politics and that not everyone may have identical motivations. So I'm sure that some of Google was pleased and relieved by this turn of events.

But mostly I wanted to share Robert's and Lisa's notes as a segue to observing that this entire issue is more a symptom than a cause and that there's an underlying problem. The cause of the actual problem is that, unfortunately, the online collection and sale of personal information has become a large, thriving, highly profitable and powerful industry. And it may be too big to stop. This is what keeps the EFF awake at night. We know enough from our previous examination of the EU's extended decision process to have seen that their decision was directly influenced by commercial interests that wanted the status quo to remain unchanged. And those interests were powerful enough to have their way. The question then becomes, how will this ever change?

The only thing more powerful than a minority of strong commercial interests is a majority of even stronger voting citizens. But people cannot dislike what they're unaware of, and the personal data collection industry has always been careful to remain in the shadows since they know how vulnerable they would be if we were to ever learn what was really going on. We've often observed on this podcast that conduct that goes unseen is allowed to occur in darkness and we know that users sitting in front of browsers have nearly zero visibility into what's taking place right before their eyes on the other side of the screen. Those who perform this spying and data collection claim that no one really cares. But the only reason people don't appear to care is that they don't really know what's going on.

When iOS began requiring apps to ask for explicit permission to track people outside of their own apps the response was overwhelmingly negative. People who were asked said no.

Similarly, it likely never occurs to the typical consumer that their own ISP who provides them with Internet bandwidth and who knows their real-world identity because they are paid every month, is in the perfect position to catch and aggregate our unencrypted DNS queries and the IP connections our account makes to remote sites. This data represents a profit center so it is routinely collected and sold. It's allowed to happen only because it goes undetected and unseen.

Nearly three years ago, in October of 2021, the US Federal Trade Commission, our FTC, published a news release with the headline: "FTC Staff Report Finds Many Internet Service Providers Collect Troves of Personal Data, Users Have Few Options to Restrict Use" and the subhead reads: "Report finds many ISPs use web browsing data and group consumers using sensitive characteristics such as race and sexual orientation."

It seems obvious that if any consumer ever gave their permission it was not truthfully made clear to them what was going to transpire. I certainly never gave my cable provider permission

to do that. But I have no doubt that the consumer agreement I originally signed but never read, or any of the updated amendments to it, contained the sort of language we've talked about before, where information about our online use may be shared with business partners and so forth.

Sometimes free enterprise can be a bit too free. This is why the protection of consumers from this sort of pervasive privacy violation for profit is the role of government. Unfortunately, government is just people too, and people can be bought. In the US at least, lobbyists for commercial interests hold a great deal of sway over how the government spends its time and our tax dollars. The US has a couple of senators who see and understand the problem. And they are investing a great deal of their time in doing what they can. But most legislators appear to feel that they have bigger fish to fry.

I think what all this means is that it is up to those of us who care, to do what we can. I'm disappointed that Google appears to have lost this round, but I understand that it probably had no choice. I'm sure we'll be talking about this again once we see what Google has come up with as their compromise.

A listener known only by their email address of "sexygenius" wrote:

Steve, there is an elephant in the room. You keep telling us "enable automatic updates" and you insist that IoT devices should be designed to automatically update themselves by default. And yet, auto-update is the reason this CrowdStrike disaster happened. Right?

If people had waited a few days before applying that CrowdStrike patch, the problem would have been minimized, right? Or even wait a few hours?

Also, people's smart TVs have recently been forced into permanent cheesy "motion smoothing" mode, and people's inkjet printers have been bricked, and Samsung Blu Ray players got bricked. All by auto-updates.

There's this big, important question that you could address: give people more granular advice regarding updates, because surely the old simplistic "always enable automatic updates" is dangerous in certain situations. Which ones? And what should we do instead?

Please dig deeper and give detailed advice for owners of IoT devices. Should we delay updates? Maybe buy two (cheap) devices and alternate between them? Should we disable auto-updates entirely for some devices? There must be some insightful and ACTIONABLE things you can say about this. Thanks Steve, -MW

And this pairs nicely with another note I received from listener Christy Ramsey under the subject "Maybe Turn On Notepad++ Updates for this one":

Notepad++ v8.6.9 bug-fixes & new enhancements:
#1. Make installation and updates easy & quiet by adding "Yes (Silent)" button.

Christy is recalling my earlier complaint that my favorite text editing Notepad replacement is Notepad++. My annoyance is that its author is constantly changing, improving and tweaking it. He never leaves it alone. And it would appear that my annoyance was not an outlier, since

Chritsy notes that v8.6.9 introduces bug-fixes and enhancements: Right. Bug fixes because its author keeps messing with it and enhancements because unless you decide that it's finished no text editor ever will be. *"Does it show the current phase of the moon down in the status bar at the bottom? No? Well what's wrong with this dumb editor?"* Apparently I'm not an outlier here since the latest new feature is to no longer be asked whether we want the new features, just silently accept them in the background.

During the final year or so of SpinRite's testing we used a local installation of GitLab for tracking and managing features and bugs. I loved it. It was great. But like Notepad++ it's apparently a project that will never be finished, since they're constantly moving it forward while cleaning up the debris they inevitably create. It's also impossible to skip any incremental version updates. So falling behind incurs the horrific penalty of needing to take the system through each incremental change that was made to get current again. And on top of this, horrifying security alerts are continually being sent out. It should not be my job nor my burden to worry about bugs in my bug tracking system. I just want to use it. So I've been determined to find a stable alternative to help manage our next work. Sorry folks, but if you refuse to leave it alone it's of no use to me.

So this brings us back to Sexy Genuis' quite reasonable question about auto updates. I've promoted the automatic autonomous updating of router firmware by their manufacturers because being the first line of defense separating our internal LANs from the increasingly hostile Internet WAN, remotely exploitable flaws really do need to be repaired before the bad guys can learn of new exploits and get in. I don't have another solution since most users of these routers have them tucked away in a closet and the last thing they're thinking about is whether the router's firmware might need updating. And they're certainly not going to be aware of the news of any newly discovered exploitable flaw.
But doesn't this make this massive network of routers potentially susceptible to a CrowdStrike style mistake where all of these routers are bricked overnight by a mistake? We've talked about this before. About ways for the router to recover to a previous stable version of firmware after a bad experience. This wasn't an option for Windows with CrowdStrike, but perhaps it should have been?

There's been ample commentary on the Internet in the wake of the CrowdStrike mess suggesting that even though CrowdStrike was the proximate cause of the trouble, from Windows standpoint the cause of the problem was an optional add-on 3rd-party device driver that did not need to be running for the system as a whole to run. So why couldn't Windows have quarantined that driver after its first crash and brought the system back online gracefully and quickly?

I think that there are not terrific answers to these problems. But they all have a common root cause, which is buggy software. And at this point in the evolution of computer systems, bugs seem to be inescapable. So we're all casting about looking for ways to better deal with bugs. And there would seem to be better answers than the ones we have so far. IoT devices and Windows should have self-healing technologies to protect themselves from mistakes made by router vendors or 3rd-parties like CrowdStrike. And that does seem like something that will happen, little by little, one earthquake tremor at a time.

Lee Mössner shared a Mastodon posting by Brian Krebs about the collection and reselling of automotive data being done by automakers without their drivers' clear knowledge or permission. Brian posted: (https://infosec.exchange/@briankrebs/112853449272107103)

> *Sen. Ron Wyden (D-Ore) has released details about an investigation into automakers' disclosure of driving data, such as sudden braking and acceleration, to data brokers for subsequent resale to insurance companies":*
>
> *"General Motors (GM) also confirmed to Wyden's office that it shared customers' location data with two other companies, which GM refused to identify."*
>
> *"The senators' letter to the FTC included new details about GM, Honda, and Hyundai's sharing of drivers' data with data brokers, including details about the payments the data broker Verisk made to automakers. Based on information Wyden obtained from automakers, the senators revealed:"*
>
> *"Hyundai shared data from 1.7 million cars with Verisk, which paid Hyundai $1,043,315.69, or 61 cents per car; Honda shared data from 97,000 cars with the data broker Verisk, which paid Honda $25,920, or 26 cents per car; Automakers used deceptive design tactics, known as "dark patterns," to manipulate consumers into signing up for programs in which driver data was shared with data brokers, for subsequent resale to insurance companies."*

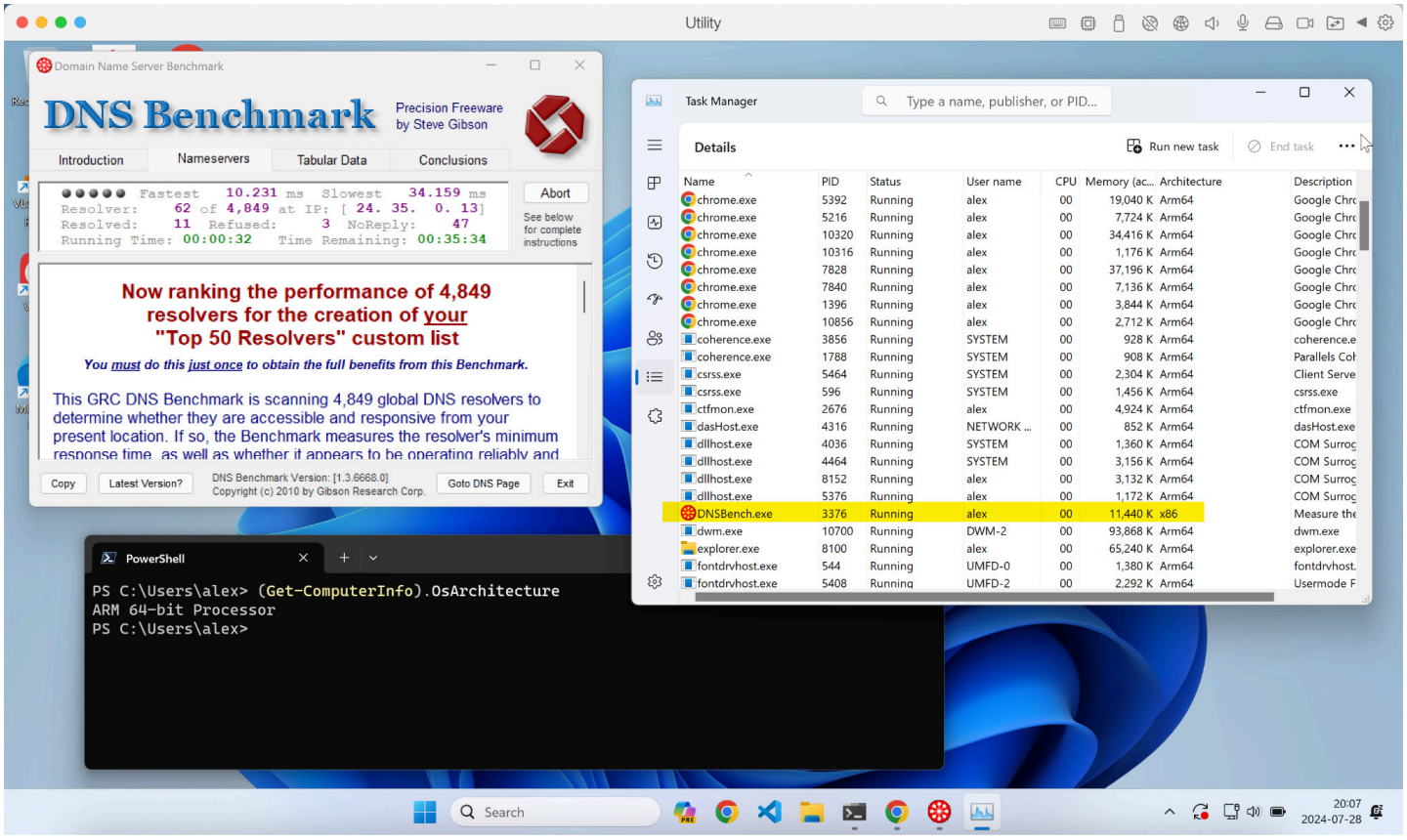
I don't have an answer to this other than for us to be aware of what's going on and take whatever measures make sense. I presume that it's no longer possible to purchase any modern vehicle that isn't connected to the Internet and dutifully feeding back everything that goes on within its perimeter to some hidden agency in the cloud. So for the time being that's part of what it means to be an auto owner and operator.

Alex Neihaus:

The podcast's friend Alex Neihaus, who was one of our earliest supporters and advertisers with the Astaro Security Gateway, sent an interesting note asking an interesting question. He wrote:

> *Hi, Steve. I tried the DNS Benchmark today running on Windows 11 ARM64 in a VM hosted on a MacBook Pro with Apple Silicon. See the image below. It appeared to run flawlessly and at full speed. (Windows 11 ARM runs faster, IMHO, on Apple Silicon than on any real PC I've tried. It's astonishing.) I'm wondering if you have an opinion about accuracy of the app's results in this scenario: emulation of x86 instructions in a VM. I think I remember you saying the DNS Benchmark is highly timing dependent for accuracy. I wonder if sheer brute computing capability, as provided by an Apple Silicon processor, can overcome the costs of double emulation. Really enjoying Security Now! these days. Thanks. Alex*

And Alex attached a screenshot of GRC's DNS Benchmark running on a Windows 11 desktop hosted on a MacBook Pro:



I included this because with the rise of ARM and Windows for ARM finally becoming real after so many false starts, I've been thinking about the fact that I write x86 code and that all of my utilities are written in Intel x86 assembly language. I've always felt that this meant that my code had unnecessary performance overkill, since it uses so very little of the processor's available resources to run. But this becomes something of an advantage when an ARM-based machine is used to emulate Intel's x86 instruction set. My utilities are objectively small because there are so many fewer instructions. And that means significantly less emulation overhead. So I think that my approach is going to stand the test of time because there's no way any version of Windows, whether hosted on Intel or ARM, will not be able to run x86 instructions one way or another, and the fewer of those there are, the faster the code will go.

So to answer Alex's question: Yes to the DNS Benchmark's proper operation under ARM. The only thing the benchmark requires for accuracy is reliable timing snapshots from the system's processor clock counter, and that would be one of the first things the designers of the x86 emulation and its VM would have provided.

Platform Key Disclosure

Today's podcast topic will reveal the details behind a widespread – and by that I mean industry wide – security failure that was discovered in the supply chain of hundreds of today's PCs from some of our largest manufacturers. The upshot of this is that these machines are unable to boot securely. And while that will be our primary focus, the larger point I hope to drive home is that this as additional evidence to substantiate my belief that Microsoft's Recall is an inherently doomed proposition – at least if we require absolute privacy for the accumulated device history that Recall proposes to aggregate. The reason for this is that despite all of Microsoft's assertions about how tightly and deeply they'll be protecting the accumulated history of their users, doing so with sufficient security is simply not possible due to the pervasive lack of security that pervades the entire PC ecosystem. Much lip service is given to how securable everything is, yet it keeps getting broken over and over again.

So what now? The security firm **Binarly** has coined the term "PKfail" for their discovery of serious problems with the Platform Keys (this PK) being used across our industry to provide the root of trust for our system's secure boot technology. Here's what they explain:

Today we disclose PKfail, a firmware supply-chain issue affecting hundreds of device models in the UEFI ecosystem. The root cause of this issue lies within the Secure Boot "master key," called the Platform Key in UEFI terminology.

The Platform Key used in affected devices is completely untrusted because it's generated by Independent BIOS Vendors and widely shared among different hardware vendors. This key is used to manage the Secure Boot databases that determine what is trusted and what instead should not be executed, effectively maintaining the chain of trust from the firmware to the operating system. Given its importance, the creation and the management of this master key should be done by the device vendors following best practices for cryptographic key management (for example, by using Hardware Security Modules). It must never be disclosed or known publicly.

However, the Binarly Research Team found that these keys are generated and embedded in a BIOS vendor's reference implementation as sample keys under the expectation that any upstream entity in the supply-chain (such as OEMs or device vendors) would replace them. When this does not happen, devices are shipped with the original "sample" untrusted keys in place. Binarly researchers identified the private part of one of these Platform Keys in a recent data dump following a leak. This key is currently being used by hundreds of devices in the market, putting every one of them at immediate risk. A peculiarity of PKfail is that it represents yet another example of cross-silicon issue, as it affects both x86 and ARM devices.

We have developed proofs of concept to demonstrate that attackers with access to a device vulnerable to PKfail can easily bypass Secure Boot by signing their malicious code and thus enabling them to deliver any sort of UEFI bootkit, like the recently discovered BlackLotus.

Modern computing relies heavily on establishing and maintaining trust, starting with trusted foundations and extending through operating systems and applications in a chain-like manner. This allows end users to confidently rely on the integrity of the underlying hardware, firmware, and software. In modern systems, trust is typically rooted in hardware-based implementations such as Intel Boot Guard or AMD Platform Security Processor.

The root trust is then propagated to the operating system via UEFI Secure Boot, which ensures that only digitally signed and verified bootloaders and OS kernels are executed by the boot manager. Secure Boot technology uses public-key cryptography and relies on four keys and databases for authentication:

1. *Platform Key (PK): The root-of-trust key embedded in the system firmware, establishes trust between the platform owner and platform firmware.*

2. *Key Exchange Key (KEK): This key establishes trust between the operating system and the platform firmware.*

3. *Signature Database (db): A database containing trusted signatures and certificates for third party UEFI components and boot loaders, which are thus granted execution.*

4. *Forbidden Signature Database (dbx): A database containing signatures and certificates used to sign known malicious software, which are thus denied execution.*

Each database is stored in its corresponding NVRAM variable (PK, KEK, db, dbx). These variables are authenticated, meaning that when Secure Boot is enabled, updates are only allowed if the update data is signed by a higher level key. The Platform Key (PK) can be used to add or remove keys from the Key Exchange Key (KEK) database, while the KEK can be used to update the Signature Database (db) and the Forbidden Signature Database (dbx).

Being at the root of the trust hierarchy, the Platform Key (PK) plays a critical role in the security of Secure Boot. Access to the private part of the PK allows an attacker to easily bypass Secure Boot: the attacker can update the Key Exchange Key (KEK) database with a malicious KEK which can be subsequently used to tamper with the Signature Database (db) and the Forbidden Signature Database (dbx). Since these databases are used during the verification process, an attacker exploiting PKfail can cause untrusted code to be run during the boot process, ***even when Secure Boot is enabled.***

The Binarly Research Team discovered that hundreds of products use a sample test Platform Key that was generated by American Megatrends International (AMI). This key was likely included in their reference implementation with the expectation that it would be replaced with another safely-generated key. That never happened!

Since these test keys are shared with commercial partners and vendors, they must be treated as completely untrusted. Several facts give us confidence in this assessment:

1. *By scanning an internal dataset of firmware images, we confirm that devices from unrelated vendors contain the same Platform Key, meaning that these keys must have been generated at the root of the firmware supply-chain.*

2. *These test keys have strong indications of being untrusted. The certificate issuer contains the clear strings: "DO NOT TRUST" or "DO NOT SHIP" in all capital letters.*

3. *More importantly, Binarly Research discovered the private component of one Platform Key in a data leak, where an alleged OEM employee published the source code containing the private Platform Key to a public GitHub repository. The private key was stored in an encrypted file, which was "protected" by a weak 4-character password and thus easily cracked with any password-cracking tool.*

Thus the untrustworthiness of this key is clear.

Shortly after discovering PKfail, it became apparent that this was not a novel vulnerability; in fact, it's quite the opposite. A quick search on the Internet returned numerous posts from users finding keys marked as "DO NOT TRUST" in their systems, worried about the security implications. But even more concerning, we discovered that the same vulnerability was known as far back as 2016 and it was even assigned CVE-2016-5247.

Why are so many devices still vulnerable to this issue almost 10 years after public disclosure?

The harsh truth is that the complex nature of the firmware supply chain -- where multiple companies contribute to the production of a single firmware image and the security of each component relies on others' security measures -- demands inspection capabilities that are far from the current industry's standards or simply unavailable from a technological point of view.

In 2019, the Linux Vendor Firmware Service project (LVFS) introduced a check based on YARA rules to detect non-production keys. This rule matches the strings "DO NOT TRUST" or "DO NOT SHIP" with the intent of identifying and reporting firmware vulnerable to PKfail. This rule works well when the Platform Key is stored in an uncompressed form, but fails when the key is compressed and stored in a raw section or within the data section of UEFI modules.

To address this and other software supply-chain security challenges, the Binarly Transparency Platform analyzes firmware images and autonomously unpacks all nested components, creating a detailed blueprint of the input, allowing for the detection of PKfail and other known and unknown security vulnerabilities.

To understand the actual scope of PKfail and its historical patterns, we scanned an internal dataset of UEFI firmware images using our Binarly Transparency Platform. This dataset is representative of the UEFI ecosystem as it contains tens of thousands of images released in the last decade by every major device vendor, including Lenovo, Dell, HP, HPE, Supermicro, Intel, MSI and Gigabyte.

***The macro results of this scan are quite alarming:** more than 10% of firmware images in our dataset use an untrusted Platform Key and are thus vulnerable to PKfail. When reducing the dataset to only more recent firmware released in the past 4 years, the percentage drops to 8%, though remaining at concerning levels.*

*The first firmware vulnerable to PKfail was released back in May 2012, while the latest was released last month, in June 2024. Overall, this makes this supply-chain issue one of the longest-lasting of its kind, spanning more than 12 years. The list of affected devices, which at the moment contains nearly **850 devices**, can be found in our BRLY-2024-005 advisory.*

A closer look at the scan results revealed that our platform extracted and identified 22 unique untrusted keys. The table below reports the five most frequently used keys, along with a breakdown of affected products and Vendors:

Certificate Serial Number	Certificate Subject	Certificate Issuer	Last Seen	First Seen	Products	Vendors
55:fb:ef:87:81:23:00:84:47:17:0b:b3:cd:87:3a:f4	CN=DO NOT TRUST - AMI Test PK	CN=DO NOT TRUST - AMI Test PK	2024-06	2018-04	364	Acer, Dell, Fujitsu, Gigabyte, Intel, Lenovo,Supermicro
-08:c2:d1:c3:6c:9b:51:4f:b3:7c:6a:02:08:12:cd:59	CN=DO NOT TRUST - AMI Test PK	CN=DO NOT TRUST - AMI Test PK	2024-06	2022-06	167	Acer, Dell, Gigabyte, Supermicro
-15:fe:0d:04:9b:3b:74:70:bc:6f:1a:d2:96:ed:c4:7b	CN=DO NOT TRUST - AMI Test PK	CN=DO NOT TRUST - AMI Test PK	2024-03	2015-01	483	Acer, Dell, Gigabyte, Intel, Lenovo, Supermicro
-1b:ed:93:e2:59:4e:2b:60:be:6b:1f:01:c9:af:a6:37	CN=DO NOT TRUST - AMI Test PK	CN=DO NOT TRUST - AMI Test PK	2023-01	2014-12	287	Dell, Fujitsu, Gigabyte, HP, Intel, Lenovo, Supermicro
1a:a9:c7:61:c8:6a:be:88:4d:85:f5:ad:2b:95:3b:f1	CN=DO NOT TRUST - AMI Test PK	CN=DO NOT TRUST - AMI Test PK	2021-03	2012-05	157	Acer, Dell, Fujitsu, Gigabyte, HP, Lenovo, Samsung, Supermicro

From the certificate subject and issuer strings, we conclude that these keys were generated by AMI. This conclusion is further supported by how these test keys ended up in devices sold by unrelated vendors, as shown in the last column of the table.

By looking at the actual product names and models, we found another concerning issue: the very same key is used to protect a highly heterogeneous set of products. For example the key with serial starting with "55:FB:EF" was found both in gaming laptops and in server motherboards. Moreover, as we can see in the Last Seen and First Seen columns, these keys survive in the ecosystem for multiple years, up to almost 10 years in case of the key with serial "1B:ED:93".

When looking at the historical trends of PKfail, several worrisome observations can be made: First, even after CVE-2016-5247 was made public, the release rate of images vulnerable to PKfail retained its previous increasing trend, suggesting that the firmware industry was not responsive to that 2016 vulnerability. This behavior is consistent with the findings from our retrospective analysis on LogoFAIL patches, where we found that the industry needed several months after the disclosure before security patches were ready and propagated to end-users.

The reaction to CVE-2016-5247 can finally be seen in the period from 2017 to 2020, where the number of vulnerable images steadily decreased. This trend however changed again after 2020 and has persisted until current day, with a constant increase of vulnerable devices, which could signal that the industry simply forgot about this problem.

Another observation related to the private Platform Key leaked on GitHub is that this leak did not result in any visible impact on the number of vulnerable devices. This is once again unsurprising when put into the historical context of the firmware industry not caring.

To be clear, this leak went almost unnoticed; in fact, we are the first to report that it contained the private part of a Platform Key. Second, the slow reaction to this security incident: by mining the data provided by the GitHub Archive and available on the Wayback Machine, we confirm that the repository remained publicly available for at least four months before getting noticed and removed by the original author, while it took five months to delete all the forks of

the offending repository. Quite concerningly, the leaked key is still in use in many devices and has been used for quite some time: the first occurrence of this key in our dataset dates back to April 2018.

So this means that while much as been made of Secure Boot, because Secure Boot is utterly reliant upon the secrecy of the private key at its root, around one out of every ten machines in use today, and even shipped as recently as last month, is only pretending to have Secure Boot because its private key is one of the handful that AMI originally provided – and clearly marked, they believed well enough – as DO NOT TRUST and DO NOT SHIP. Yet shipped they were and are still being.

One very cool bit of tech that Binarly shared is the way any Linux or Windows user can check their own PC(s) for the presence of these insecure keys. They wrote:

Devices affected by PKfail will have the strings "DO NOT TRUST" or "DO NOT SHIP" in the subject and issuer fields of the Platform Key certificate. On Linux, PKfail can be easily detected by displaying the content of the PK variable:

```
$ efi-readvar -v PK
Variable PK, length 862
PK: List 0, type X509
    Signature 0, size 834, owner 26dc4851-195f-4ae1-9a19-
fbf883bbb35e
        Subject:
            CN=DO NOT TRUST - AMI Test PK
        Issuer:
            CN=DO NOT TRUST - AMI Test PK
```

On Windows, running the following command in a privileged PowerShell console will return True on affected devices:

```
> [System.Text.Encoding]::ASCII.GetString((Get-SecureBootUEFI
PK).bytes) -match "DO NOT TRUST|DO NOT SHIP"
True
```

The Binarly Research Team recommends that affected users update their firmware when device vendors release an updated version with fixes for PKfail. Expert users can re-key the PK with a trusted key. The other Secure Boot databases – KEK, db and dbx – must also be assumed compromised, thus expert users should check and replace them with new databases of trusted signatures and certificates.

Just so everyone is clear, there is no obvious remote vulnerability here. The primary danger is from local boot tampering with a machine that was believed to be secured against exactly such manipulation. That said, we've seen plenty of instances where remotely injected instances of malware were then able to then establish persistent bootkit rootkits by messing with the user's motherboard firmware from software.

So this is not the end of the world by any means. And since this time Binarly's research, which is somewhat breathtaking in its scope, has generated far more interest than the issue did back in 2016, and since there really does appear to be a great deal more security awareness today than eight years ago, I would expect that the manufacturers of all still supported systems will arrange to respond to the egg that they all now have on their faces. There really is no excuse for being so negligent as to ship systems whose master Platform Keys are screaming DO NOT TRUST and DO NOT SHIP in all caps.

I'm not going to tell anyone that they should not use Recall once it becomes available. That's not my place. I understand 100% that the reward from its use may well justify the very slim chance that the data it gathers might be put to some use that its owner would find objectionable. After all, that's certainly happening no matter where we go today on the web or even where and how we drive our cars in the real world. Why should this be any different?

But that said, what does appear to be imperative, Microsoft's repeated and well-meaning assertions about their ability to secure this information notwithstanding, for every individual to be given the well informed choice about whether or not they want this for themselves and for that choice to then be honored without exception or excuse.