Windows Platform Binary Table

Description: This week we're back to answer a collection of burning questions which we first pose, including: What news from HP? What is Microsoft doing for Windows 11 that promises to break all sorts of network connections? What's OWASP's new Top Ten list of worries about? Did Apple help the NSA attack the Kremlin, and what crucially important revelation does this incident bring? What new hacking race has Google created? And what misguided new U.S. legislation will hopefully die before it gets off the ground? What is TOR doing to protect itself from DoS attacks? How much are educational institutions investing in cybersecurity? And what can go wrong with civilian cameras in Ukraine? Are we seeing the rise of cyber mercenaries? What is the Windows Platform Binary Table, why should we care, and how can we turn it off?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-926.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-926-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Coming up, OWASP's suggestions for things to watch out for when you're using artificial intelligence. Did the NSA and Apple work together to hack the Kremlin? Steve doesn't think so, but the story's quite interesting. And then we'll talk about the Gigabyte motherboard flaw that could infect hundreds of millions of people. Yup, Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 926, recorded on Tuesday, June 6th, 2023: The Windows Platform Binary Table.

It's time for Security Now!, the show where we cover the latest security, and the security news, and how things work, and laugh a little, and love a little. And no hugging. Mr. Steve Gibson.

Steve Gibson: And why not?

Leo: And why not. Steve from GRC.com, longtime host of this show. Great to see you.

Steve: And, you know, I had an occasion to go back to 2015, we'll get to that in a while, in one of - I think it was Episode 542 or something. And I was reading the transcript, and I thought, you know, I am really glad that this has evolved into a script that I'm reading because I'm not good when it's with just completely freeform, and I'm just like...

Leo: Oh, is that how we used to do it, where I would like just - we would just gas about stuff?

Steve: Well, just like all the other podcasts that you have.

Leo: Oh, yeah. Yeah, that's true.

Steve: Yeah. But, you know, but you guys are good. I'm stumbling around, and I'm like, my god. And Elaine had to transcribe this. I'm sorry, Elaine. Anyway, we've got a great one today. This one is titled Windows Platform Binary Table. And I loved your little quip earlier, like random words that you have to memorize to show that your brain is still functioning. It sounds a little bit like that.

But we're going to start off by answering a bunch of questions. What news from HP? What is Microsoft doing for Windows 11 that promises to break all sorts of network connections? What's OWASP's new Top 10 list of worries about? Did Apple help the NSA attack the Kremlin? And what crucially important revelation does this incident bring? What new hacking race has Google created? And what misguided new U.S. legislation will hopefully die before it gets off the ground? What is TOR doing to protect itself from DOS attacks? How much are educational institutions investing in cybersecurity? And what can go wrong with civilian cameras in Ukraine? Are we seeing the rise of cyber mercenaries? And what is the Windows Platform Binary Table? Why should we care? And can we turn it off, please?

Leo: Ooh. Well, I look forward to this episode of person, woman, man, camera, TV. See, I remember them. I remember them.

Steve: That's good. And we do have another Picture of the Week, and it's one that we've seen many times now, but this one sort of pushes it over the top. So it made the grade mostly because of how way more than normally ridiculous it is.

Leo: I haven't looked at it yet. We have a new protocol. I don't look at the Picture of the Week until the actual time so you can see my...

Steve: The listeners can share your experience.

Leo: ...natural reaction; that's right.

Steve: That's right.

Leo: All right. Steve, I'm ready. My eyes' blinders have been removed. This is the Picture of the Week. I'm going to scroll it up, and then I'm going to look over and see it. Are you ready? Here we go. The Picture of the Week [laughing]. That is as bad as you can get.

Steve: So, yeah. The reason - so we're being - I have enough pictures of ridiculous gates...

Leo: I bet, I bet.

Steve: ...in the middle of, like, fields and things, that we'll have those for the end of time. And we've had so many of them recently that I thought, well, I'm not, you know, we're going to back off on that theme a little bit. But this one just - it made it to the podcast because it goes the extra mile. It's, you know, it's - so we have this path, like, some steps are going down in the foreground, and there's a path cutting across them going into the distance. And there's this gate which has been placed on the path. And it's not just that there's a gate there, which of course you could just walk around, because like choose which way - it's probably a little more convenient to go around it onto the right because the hill gets a little steep descending to the left.

Leo: It's not about a gate, it's just a sign holder, basically.

Steve: So what this thing has the audacity to say to somebody who is confronting it is, in red signage: "Strictly No Access."

Leo: Oh, no, absolutely not.

Steve: No, really, strictly none, despite the fact that, like, what? You could just walk around it.

Leo: There's also, I'm sure you noted this, there's a - is that a power outlet or what is that? I mean, it seems like they maybe electrified the fence.

Steve: That was my thought, or that they're monitoring it in case somebody opens it and goes through.

Leo: Don't you dare. That is the weirdest thing ever.

Steve: And the only thing I could think was that this was there so that, if somebody was caught on the path behind the gate, they could say, well, you were warned.

Leo: You were warned.

Steve: Strictly no access. Anyway, I just - this was just a hoot.

Leo: That's hysterical.

Steve: Because it's like, you know, pushes this thing really over the top. Right. Not, I mean, the gate being there would be enough. But no, you really - it doesn't matter how much you want to go down that path. No. Can't.

Leo: Oh, lord.

Steve: Okay. So I started off this week, as I do every week, looking to see whether there was any good news for the tens of thousands of owners of HP OfficeJet 9020e series inkjet printers which were all bricked, as we know, that Monday morning, now four weeks and a day ago. Again, total silence from HP. Against that backdrop, the first two headlines that popped up in my search yesterday for any updated news were stories initially published when this was news was fresh: "HP rushes to fix bricked printers after faulty firmware update," and "HP Races to Fix Faulty Firmware Update That Bricked Printers."

So we have "rushing" and "racing." Both of those and many similar headlines certainly made sense at the time, but they're not aging well. Despite what we hope and assume from HP, their response, even after now more than a month, has been a big zero. So, boy. Wow. Just it's crazy. And again, this wasn't a small event. This didn't happen to a few people. This was global. And I refer to it a little bit later in the show.

Okay. So Microsoft has announced that future versions of Windows 11 will require all SMB, as in Server Message Blocks, messages to be cryptographically signed, regardless of the message type. Now, that's a rather significant change, and it's going to be interesting to see what it breaks. SMB is commonly known, it's the protocol that many things in Windows now use. Notably, Windows File and Printer Sharing runs on top of Server Message Blocks. The requirement for cryptographic signing is different from, and a further extension of, Microsoft's policy of stepping away from the oldest versions of this aging and sadly troubled protocol.

Just over a year ago, Microsoft announced their related intention to remove all SMBv1 support from Windows 11. Back then, at that time, Ned Pyle, the Microsoftie who was making this announcement into Microsoft's Tech Community, he wrote: "I had to save this behavior for last. It's going to cause consumer pain among folks who are still running very old equipment, a group that's the least likely to understand why their new Windows 11 laptop cannot connect to their old networked hard drive." He said: "I'll spread the word throughout consumer channels as best I can. I hope you can help me with friends and family who rely on you for technical expertise."

Okay. So Microsoft is, you know, changing things, and they feel that they need to. There are currently three major versions of SMB - 1, 2, and 3. And actually I think it's 3.1.1 currently. And by default, Windows 10 has always disabled support for the original SMBv1 due to its total lack of security, but the code to support both the client and the server roles has remained present in Windows 10. Remember that SMBv1 was present in the MS-DOS Windows for Workgroups add-on which allowed DOS machines, which were still very much in use in the early days of Windows, to participate as either clients or servers on a Windows LAN.

Since my own work on SpinRite 6.1 is still DOS-based, and DOS only knows about the original SMBv1, I was hugely relieved to find that support for v1 could be enabled under Windows 10. That's what I've been using to network my DOS machines. It's allowed me to write and assemble SpinRite's code on a Windows 10 workstation, then give the DOS machine access to that Windows 10 shared directory which contains SpinRite's source and executables so that I'm able to do native source level debugging under DOS.

Last year Microsoft took this further by removing the binaries for SMBv1 from at least some of Windows 11. So it was disabled under Windows 10, but you could go under that - it's under Control Panel. Oh, it's Windows Features where you're able to, like, turn things on and off, like for example a web server. You're able to say, yeah, I'd like to have an IIS web server on my Windows workstation, and turn it on, and it sort of adds that to Windows. So that's where this SMBv1 stuff is in Windows 10 that you can still turn on. Microsoft is, you know, needing to keep moving things forward. So they've decided that you're not even going to be able to that in the future.

And apparently this still breaks things, that is, removing SMBv1. So users of the higher end editions even of Windows 11 will be able to use an unsupported binary from Microsoft to add this back. I'm just telling everybody so in case this happens to you, and for some reason you still need v1. You know, there are things, like I think the version of the Sonos software that I had wasn't able to operate unless it had v1. So then there were instructions for how to go turn that on.

But anyway, this next announcement will be interesting, and it's likely to create further issues for connectivity with devices that cannot or do not support SMB message signing because that's the news of the week. And that makes me suspect that this signing requirement is also likely to be somewhat soft and probably can be overridden because Microsoft's just - they're really reluctant to break things that have happened before.

So last Friday that same guy, Ned Pyle, who a year ago said what I first quoted, last Friday he said: "Heya, folks. Ned here again. Beginning in Windows 11 Insider Preview Build 25381, the Canary release, Enterprise editions, SMB signing is now required by default for all connections. This changes legacy behavior, where Windows 10 and 11 required SMB signing by default only when connecting to shares named SYSVOL and NETLOGON, and where Active Directory domain controllers required SMB signing when any client connected to them."

So SMB signing is a simple but a useful security measure that's been around since actually, believe it or not, Windows 98 had it, and Windows 2000 over on the NT platform. But it's never been forcibly enabled by default because the signing overhead it adds to every message used to be too much. Without signing, nothing detects or prevents the alteration or spoofing of SMB messages. There's no security in them otherwise. So signing protects against NTLM, NT LAN Manager, relay attacks which have been a constant thorn in Microsoft's SMB implementation. So signing is definitely a good thing.

But, as always, moving into the future for the sake of security also means removing some of the past, which is always somewhat painful. I'm mentioning all of this as a heads-up since there's a very good chance that this next move by Windows 11, and it sounds like it may be only Enterprise editions, so it may not be affecting Pro users and Home users, which is probably a good thing. But even at the Enterprise it could break some things that our listeners are using, you know, things that either don't currently have SMB signing enabled or maybe don't support SMB message signing at all. So there.

OWASP, O-W-A-S-P is the "Open Web Application Security Project." And it has, for the past 20 years, since 2003, maintained the often quoted "OWASP Top 10" list of the most worrisome web application vulnerabilities. I have a special warm spot for OWASP, since it was several European chapters of OWASP who hosted my trip to Europe once SQRL was completed to give me the opportunity to introduce it to their members in person, which I really appreciated. We're talking about this today because OWASP has announced their work on a completely different list. It is the "OWASP Top 10" for Large Language Model applications, which is interesting that they've decided, okay, there's enough issues with LLM stuff that they're going to start tracking and maintaining a similar Top 10 list of things to be careful about.

So in their announcement they said: "The OWASP Top 10 for Large Language Model Applications project aims to educate developers, designers, architects, managers, and organizations about the potential security risks when deploying and managing Large Language Models. The project provides a list of the Top 10 most critical vulnerabilities" - and actually it's vulnerability classes - "often seen in LLM applications, highlighting their potential impact, ease of exploitation, and prevalence in real-world applications. Examples of vulnerabilities include prompt injections, data leakage, inadequate sandboxing, and unauthorized code execution, among others. The goal is to raise awareness of these vulnerabilities" - again, vulnerability classes - "suggest remediation strategies, and ultimately improve the security posture of LLM applications."

They said: "The following is a draft list of important vulnerability types for Artificial Intelligence applications built on Large Language Models." So we've got 10 itemized problems or classes. The first is prompt injections. "Prompt injections bypass filters," they write, "or manipulate the LLM using carefully crafted prompts that make the model ignore previous instructions or perform unintended actions." And we have already seen applications of that.

Number two, data leakage: "Accidentally revealing sensitive information, proprietary algorithms, or other confidential details through LLM responses." Number three, inadequate sandboxing: "Failing to properly isolate Large Language Models when they have access to external resources or sensitive systems, allowing for potential exploitation and unauthorized access."

Fourth, unauthorized code execution: "Exploiting LLMs to execute malicious code, commands, or actions on the underlying system through natural language prompts." Fifth, SSRF, Server Side Request Forgery Vulnerabilities: "Exploiting LLMs to perform unintended requests or access restricted resources, such as internal services, APIs, or data stores."

Number six, over-reliance on LLM-generated content: "Excessive dependence on LLM-generated content without human oversight can result in harmful consequences." You're right. Seven, inadequate AI alignment. They said: "Failing to ensure that the LLM's objectives and behavior align with the intended use case, leading to undesired consequences or vulnerabilities." Number eight, insufficient access controls: "Not properly implementing access controls or authentication, allowing unauthorized users to interact with the LLM and potentially exploit vulnerabilities."

Nine, improper error handling: "Exposing error messages for debugging information that could reveal sensitive information, system details, or potential attack vectors." And finally, 10, training data poisoning: "Maliciously manipulating training data or fine-tuning procedures to introduce vulnerabilities or backdoors into the Large Language Model."

Okay. So some of these are old and generic, like unauthorized code execution, you know, that's not something anyone is going to want, you know, except bad guys. And similarly, data leakage, inadequate sandboxing, server-side request forgeries, insufficient access controls, or improper error handling. You know, those are like, yeah, right, sort of common generic classes of problems. But those common problems are on this list because this is a new context, and there might be some tendency to let those slip by, thinking that those old rules no longer apply. So I agree that it's certainly worth reinforcing and remembering to think specifically about some of those oldies but goodies. There's a reason they have become so well known, and they never quite disappear.

And then we have a handful of new problems that are quite specific to this application class. "Prompt Injection" is a new one. That didn't exist before we had conversational chatbots, where we quickly learned that you could lead them astray. You could get them to do things that they weren't wanting to do. In one of our very first conversations about

this we were talking about how some bad guys managed to make one of the early ChatGPTs do something, and it sounded like they did it just by like asking more and getting mad at it. And finally it said, yeah, oh, oh, yeah, okay, I'll give you what you need. So there's that.

We have "Overreliance on LLM-generated Content," which as we have seen examples, we talked about one - I guess it was here, or maybe it was just, you know, other TWiT podcasts - where an attorney used ChatGPT to generate a complete legal brief, citing cases which were fictitious. It just made them up. So, yeah, overreliance on LLM-generated content. We've got "Inadequate AI Alignment" and "Training Data Poisoning." So in addition to all of the usual suspects, we also have a few new worries that we've never had before.

The Original OWASP Top 10 has for 20 years been a useful benchmark against which many projects have been measured and just checked. If nothing else, coders of any stripe, you know, back then and for the last 20 years, have been able to use it to double-check that they hadn't overlooked some obvious things. I expect that this LLM-specific Top Ten list will serve a similar role. Someone should create one. OWASP is at work doing so. And I think they said that they were at 0.1 at this point, so not yet ready for primetime. But they have announced that they're going to be working on this.

Leo: It looks pretty good. I mean, I would agree with every single one of those. They all look pretty accurate.

Steve: Yeah, yeah. And, you know, it's so easy just to deploy these things without giving sufficient thought to things that you should give sufficient thought to.

Leo: Yeah, yeah.

Steve: So good to have it. Okay, now, this is a two-parter. And this ended up with something really interesting, an observation that we're making for the first time anywhere. And it's, I think, very important. So we start off with what led me into this, which was the question, did Apple help the NSA attack the Kremlin? It should have been given the subheading "Why does anyone take anything Russia says with even a grain of salt?" Now, I was unable to read the original - I've got the link in the show notes, but it's HTTP, right, because Russia doesn't believe in "S" - :// and then www.fsb.ru, and the rest is in Russian. And it didn't seem worth bothering to get it translated because I was able to use the translation from the Risky Business newsletter. So I'm going to rely on that. And here's what Risky Business explained.

"Russia's FSB intelligence service claims to have uncovered a U.S. intelligence operation that hacked the Apple smartphones of diplomatic missions and embassies in Russia. The operation allegedly targeted thousands of devices, including the devices of Russian citizens and diplomatic representatives from NATO countries, the post-Soviet bloc, Israel, China, and South Africa. The attacks exploited a vulnerability in Apple smartphones. The FSB attributed the hacks to the U.S. National Security Agency and claimed Apple cooperated with the NSA for the attacks." And actually it was that allegation that caught my attention since, well, that would be huge if it turned out to be true, which seems unlikely in the extreme to me, given everything we know about Apple, and fortunately the country that we still live in.

Anyway, Risky Business continues, saying: "The Russian cybersecurity firm Kaspersky says the same attacks which the company tracks as Operation Triangulation also

targeted its employees. Kaspersky said they found compromised devices as far back as 2019 and said that the attacks are still ongoing. According to Kaspersky and to a technical report released by FSB's National Coordination Center for Computer Incidents, the attacks involve an iOS zero-click exploit delivered as a file attachment via iMessage. The attachment executes without user interaction as soon as it arrives on a device and starts downloading additional files. Kaspersky described the final payload as a 'fully-featured APT (Advanced Persistent Threat) platform.' Unlike the FSB, Kaspersky did not link the activity to the NSA or any other APT group, and could not say if the attacks targeted other organizations.

"News of the attacks came after, in March, the Kremlin's security team instructed their presidential staff to dump their iPhones by April 1st, 2023." Coincidentally April Fool's Day. And we talked about that Kremlin activity at the time on the podcast. "Employees of the Kremlin were told to get an Android device, either from a Chinese vendor or one running Rostelecom's Aurora OS. Kremlin officials cited security considerations for their decision, claiming iPhones were 'more susceptible to hacking and espionage by Western experts compared to other smartphones.'"

Leo: Which is not true.

Steve: Which is not supported by any evidence we've seen. "Russian officials asked the Prosecutor General's Office to start a formal investigation into Apple employees and U.S. intelligence officials." Okay, right. Okay. And, unsurprisingly, in an email that the Risky Business newsletter received after its initial publication of this news, Apple formally denied the FSB's accusations, writing: "We have never worked with any government to insert a backdoor into any Apple product, and we never will." And as I said earlier, given the entire history of Apple's actions and the design of their devices which we have often examined as closely as we could, I certainly believe Apple's assertion far more than the Kremlin and the FSB who we catch frequently spewing state-sponsored propaganda.

Okay, now, that said, if Kaspersky first saw this attack as early as 2019, assuming that the same zero-click exploit was in use then as is in use now, that would suggest that an exploit has remained undiscovered for the past four years. So I was interested in additional details, and something that an actual security firm did in terms of research. So I tracked down Kaspersky's thoughts about this.

They wrote: "While monitoring the network traffic of our own corporate WiFi network dedicated for mobile devices using the Kaspersky Unified Monitoring and Analysis platform" - which they call KUMA, K-U-M-A - "we noticed suspicious activity that originated from several iOS-based phones. Since it is impossible to inspect modern iOS devices from the inside, we created offline backups of the devices in question, inspected them using the Mobile Verification Toolkits, and discovered traces of compromise.

"We are calling this campaign 'Operation Triangulation,' and all the related information we have on it will be collected on the Operation Triangulation page. If you have any additional details to share, please contact us at triangulation@kaspersky.com. Mobile device backups," they wrote, "contain a partial copy of the filesystem, including some of the user data and service databases. The timestamps of the files, folders, and the database records allow the rough reconstruction of the events happening to the device. The mvt-ios utility produces a sorted timeline of events into a file called 'timeline.csv,' similar to a super-timeline used by conventional digital forensic tools.

"Using this timeline, we were able to identify specific artifacts that indicate the compromise. This allowed us to move the research forward, and to reconstruct the general infection sequence. One, the target iOS device receives a message via the

iMessage service, with an attachment containing an exploit. Two, without any user interaction, the message triggers a vulnerability that leads to code execution."

Leo: A zero-click.

Steve: Yes. "Third, the code within the exploit downloads several subsequent stages from the command-and-control server that include additional exploits for privilege escalation. Fourth, after successful exploitation, a final payload - a fully featured APT platform - is downloaded from the command-and-control server. And fifth, both the initial message and the exploit in the attachment are deleted."

They said, finishing: "The malicious toolset does not support persistence, most likely due to the limitations of the OS." Or actually the strength of the protections that iOS is giving us. So it's living in RAM, no persistence. "The timelines of multiple devices indicate that they may be reinfected after rebooting. The oldest traces of infection that we discovered happened in 2019." Okay, now, so that's a correction to what my initial presumption was. I had presumed they first saw this in 2019. No. In looking back at forensic evidence that was being collected, the oldest traces of the infection they discovered in their records occurred in 2019.

"As of the time of writing," they said, "in June 2023" - so that's now - "the attack is ongoing, and the most recent version of the devices successfully targeted is iOS 15.7." So also current iOS. Well, almost current. "The analysis of the final payload is not finished yet. The code is run with root privileges, implements a set of commands for collecting system and user information, and can run arbitrary code downloaded as plugin modules from the command-and-control server."

Okay. So one of the things that gives this apparently long-running attack campaign so much power and longevity is, as Kaspersky wrote: "It is impossible to inspect modern iOS devices from the inside, so we created offline backups of the devices in question." To that observation we add the fact that this exploit fully covers its own tracks by deleting the exploitive attachment and the original attachment-carrying iMessage.

Now, add to that the fact that iMessage is end-to-end encrypted using private decryption keys that are only present in each of the endpoint devices' secure enclaves, and that these attacks are all individually targeted at their victims. That means that communications traffic monitoring cannot be used since all anyone on the outside, like Kaspersky, who wishes they could see what was going on, will ever see is pseudorandom noise flowing back and forth. Taken together, what all this means from a practical forensics and remediation standpoint is that this thing can never be caught.

And that brings up an interesting point that has never been observed during the 17-plus years of this podcast, which is, to exactly the same degree that Apple's seriously super-strong security is protecting the privacy of its users, it is equally protecting the privacy of exploits like this from discovery. Now, as we know, the Pegasus exploits are left behind in their targets' phones to later be discovered, reverse engineered, patched, eliminated, and rendered inert. But even as skilled a forensics team as Kaspersky can only observe an historical log of file modifications made by iPhone backups that indicate that something may have been happening to them, and is happening now, and for the past four years, with no idea by whom or to what end. And no one, Kaspersky or anyone else, can take this any farther.

So I'll reiterate, since it seems like an important observation: To exactly the same degree that Apple's seriously super-strong security is protecting the privacy of its users, it is

equally protecting the privacy of exploits like this one from ever being discovered and eliminated.

Leo: You know, it's funny, I saw that same sentence when I read the story days ago. I didn't put two and two together. It's an interesting problem, though. But, I mean, you want them to have encrypted backups; right?

Steve: Oh, yeah. I'm not saying that this is bad.

Leo: It's not a bad thing, but it does prevent the discovery of the provenance. Or removing it, yeah.

Steve: Yes. And so the difference here is that this thing can delete itself. And apparently the Pegasus exploits don't have that ability, or they certainly would.

Leo: Right.

Steve: You know? They get left behind, and then we can figure out who got attacked and what the attack was and what the exploit is, and we reverse engineer it, and Apple gives this an emergency iOS update so that none of us can have that happen to us, even though it's unlikely to because it's targeted. This thing is four years in duration. And it's, like, someone's got it. Someone's using it.

Leo: Right.

Steve: And iOS is protecting it.

Leo: Now, if the victim unlocks the phone for the forensic analysis, couldn't that make it possible? Or no?

Steve: The whole thing happens while the phone's locked.

Leo: If it's gone already, there's nothing you can do; right?

Steve: Right. It's received, executes, downloads some stuff...

Leo: And erases itself.

Steve: ...and immediately deletes itself.

Leo: Which is, by the way, a tale as old as time. I mean, that's what hackers have always done is erase their tracks.

Steve: Right. You immediately remove your own intervention from the logs in classic computer attacks.

Leo: Right, right.

Steve: But anyway, so something is out there.

Leo: Scary.

Steve: It is. That's, yeah. And it can go and get anybody it wants, is what this amounts to. Anybody whose phone or iOS account is known can receive an iMessage and, in the blink of an eye, this thing is in and out and leaves behind a running advanced persistent threat alive in their phone until it's restarted that has apparently free rein.

Leo: Apple could implement some form of logging that would make note of this and not be addressable by the hack. I would think.

Steve: Yeah, I would think that's true.

Leo: Yeah, some sort of syslog that is, you know, encrypted and protected. Maybe they should. I mean, that would solve it; right?

Steve: Or maybe, I mean, so do they know about this? It's been going on for four years.

Leo: Well, it's in the news.

Steve: Yeah, I mean, they know now.

Leo: Oh, you're thinking as Russia is asserting that they do know about this. And they're intentionally...

Steve: No, no, no. I still believe, I mean...

Leo: Apple wouldn't do that.

Steve: I don't think they would. Everything we know about them says that they - sorry, no, NSA, you're going to have to find your own way in. And I can't [crosstalk]...

Leo: If they're not going to help the FBI, they're not going to help the KGB.

Steve: Now, the way this might go away is just naturally. There could be some new features added to iMessage in iOS 17.

Leo: Right, right. That just happened to fix it.

Steve: Yes, exactly. They do a recompile, or they reimplement it, and it breaks the thing that they were using for so long.

Leo: Right.

Steve: On the other hand, how many iOSes have we had over the last four years, and this thing apparently has stuck around across multiple iOS major version changes.

Leo: And it's often in messages or the browser that you find these exploits because they have rendering engines that are basically code engines.

Steve: Right.

Leo: And we've talked about that before. It seems like every operating system should have some sort of forensic log that is permanent and unmodified.

Steve: Immutable, yeah.

Leo: Yeah, immutable forensic log. But maybe that's too much to ask. I don't know. You know, it certainly would be a burden on resources. That's interesting.

Steve: These things have so much RAM now.

Leo: Yeah, we got us some resources. I think my phone, frankly, is barely in use. I mean, it's just sitting there waiting to do something.

Steve: Yeah, you know, it may be the targeted nature of it, the fact that it deletes itself, you know, I mean, and like what, you know. And so Apple just doesn't have the opportunity to get their hands on this. Or I don't know. Really, really, really interesting, though.

So last Thursday Google announced what they called their "Chrome Browser Full Chain Exploit Bonus" program.

Leo: Woohoo. That sounds good.

Steve: Yeah. Yeah, it's a bonus. Here's what they said. They said: "For 13 years, a key pillar of the Chrome Security ecosystem has included encouraging security researchers to

find security vulnerabilities in Chrome browser and report them to us through the Chrome Vulnerability Rewards Program, the Chrome VRP.

"Starting today" - that was last Thursday, June 1st - "until December 1st, the first security bug report we receive which provides a functional full chain exploit resulting in a Chrome sandbox escape, is eligible for triple the normal full reward amount."

Leo: Woohoo.

Steve: Your full chain exploit could result in a reward up to $180,000.

Leo: Yikes.

Steve: And potentially more with other bonuses. And because I didn't want people to stop after the first one, or to be discouraged, they said: "Any subsequent full chains submitted during this time are eligible for double the full reward amount. So $120,000 each." So essentially Google's creating an extra incentive race among bug hunters. Anything found before this coming December yields - anything found yields double the normal bounty, and the first person to supply an unknown exploit is rewarded for their trouble with triple the normal payment.

So they said: "We've historically put a premium on reports with exploits" - they call them "high quality reports with a functional exploit" - "is the highest tier of reward amounts in our Vulnerability Rewards Program. Over the years, the threat model of Chrome browser has evolved as features have matured, and new features and new mitigations, such as the MiraclePtr, have been introduced. Given these evolutions, we're always interested in explorations of new and novel approaches to fully exploit Chrome browser, and we want to provide opportunities to better incentivize this type of research. These exploits provide us valuable insight into the potential attack vectors for exploiting Chrome, and allow us to identify strategies for better hardening specific Chrome features and ideas for future full-scale mitigation strategies."

So, you know, 180 grand, that's not chicken scratch. And if you're a competent reverse engineer hacker researcher person, there's an extra year's worth of income for you in however long it takes you to find something if you can be first. And even if not, as long as what you find is unique, 120 grand. So anyway, I've got a link in the show notes with the full participation details for anyone who might be interested.

Okay. Oh, boy. Exactly one week ago, last Tuesday, three quite conservative senators who belong to the Republican party introduced their "Know Your App Act." Their announcement carries the headline "Wicker, Scott, Lankford Introduce Bill to Increase Transparency [and of course] Better Protect [the] Children Online."

The announcement of this new legislation begins: "WASHINGTON - U.S. Senators Roger Wicker, Tim Scott, and James Lankford introduced the Know Your App Act. The bill would require online app stores to display the country where apps are developed and owned." So this is, I suppose, the natural follow-on from all the controversy surrounding TikTok, once it became clear to them that TikTok was not a breath mint. They then go on to explain this new proposed legislation's intent, starting with quotes from each of the three senators.

Roger Wicker leads with his quote: "Our adversaries will exploit every available tool, including popular apps that gather huge amounts of data on Americans, to gain an

advantage over the United States. It is crucial for users to take steps to limit their exposure and be made aware of the risks associated with using foreign-controlled apps." I know. It gets worse, Leo.

"The Know Your App Act would bring much-needed transparency to app stores, empowering Americans to safeguard their families from exploitation." You know, from the Commie bastards. Wow. Then we hear from Tim Scott: "Americans should be able to make informed decisions about the online services they use in order to protect their data and security. Requiring app stores to display an app's country of origin is a commonsense solution that can help them do just that. Parents shouldn't fear that their family's online privacy and security could be compromised when unknowingly using an app owned by a foreign adversary."

And finally, James Lankford adds: "Seeing 'Made in China' on nearly any product nowadays is frustrating to Oklahomans" - he's a Republican Senator from Oklahoma - "frustrating to Oklahomans trying their best not to prop up the Chinese Communist Party and Chinese government with their hard-earned money. We already see the ways the TikTok app is a dangerous extension of the CCP that is collecting every user's personal data and all of their contacts. I want the 'Made in China' label and labels for any other countries where apps like TikTok originate to be clearly marked when and where they are downloaded. Americans should remain free to buy items from wherever they want."

Leo: Please.

Steve: I know, "But the least Big Tech can do is label where Americans' money" - these are all free, by the way - "is going when they download in the app store." Okay, now, aren't all iPhones and iPads...

Leo: Made in China, yeah.

Steve: ...and pretty much iAnything also made in China?

Leo: Although they, by law, they do have to put on it Made in China. Anything made in China by law has to be labeled that way. The problem is that software development's not like manufacture. Most programs are written all over the world by a variety of teams; right?

Steve: Yeah. You know, and the guts of our cars.

Leo: Oh, yeah.

Steve: It's why that COVID-related Chinese chip shortage messed up U.S. and foreign automobile manufacture and jacked up the cost of used cars that already had all their chips built in. So but it's worse. The legislative announcement continues with: "As of March 2023, four of the five most popular apps in the U.S." - this is the proposed legislation from last Tuesday - "four of the five most popular apps in the U.S. were developed in China. This is particularly concerning given that China's national security laws provide a pathway for the Chinese Communist Party to compel application developers to control an application's content or user data. The Know Your App Act

responds to this risk by requiring online app stores to display prominently the country where apps are developed and owned, allowing users to make informed decisions about the applications they access."

And then it gets worse: "The bill also requires the U.S. Department of Treasury and U.S. Department of Commerce to produce a list of adversarial governments that may have undue control over application content moderation, algorithm design, or user data transfers. App stores would be required to provide users the ability to filter out applications from the identified adversarial countries and" - get this - "warn users about the risk of downloading one of the foreign applications on these lists. If a developer fails to provide sufficient information to the app store about its country affiliation, the app store would be required to issue multiple warnings over a designated period. If the developer still refused to comply, the app store would be required to remove the app from its store." Says the party that doesn't want the government to be involved in stuff.

Leo: I mean, this is minor. I guess it's possible, and it's certainly not a hardship, I would guess.

Steve: Well, yeah, I mean...

Leo: It's meaningless. It's not going to change anything.

Steve: Right. So products, as we know, as you said, Leo, are routinely marked with their country of origin. So that's not any new big deal. But here we're implicitly saying that any applications made in China are inherently dangerous for that reason, which is what really feels wrong to me. And it's not going to work anyway. If someone wants Temu, TikTok, CapCut, or Shein, those are the top four or five. Meta's thing I'm blanking on right now is the fourth of the top five. Instagram. That rounds out the top five. If they want those apps, that's what they're going to download.

No one who needs to use these apps for their intended purpose is going to care where they came from. And the idea of requiring an app store to caution and warn a user that an app was developed in China with an "Are you sure?" before it can be downloaded seems to me really unfair. Perhaps I'll be proven wrong in time, but today this seems really over the top. Increasing the tension and division between two of the world's superpowers doesn't seem like a winning strategy for anyone. Wow. Made in China for apps.

Okay. The Tor Project is testing a new Denial of Service mitigation feature for their network, where servers will require their connecting clients to solve a puzzle to access its resources. So, you know, it's basically a CAPTCHA; right? Now, it won't normally be enabled, but it will become active when a server is being overloaded with bogus attack requests. The idea is to allow authentic users to connect to a Tor service while weeding out automated DDoS attacks, which in recent times have become a big problem for Tor servers.

The new feature is currently being tested in the Tor alpha software and is expected to roll out to most Tor nodes later this year. Work on the feature started last year after Tor node operators reported DDoS attacks against their infrastructure. And, you know, probably hacker kiddies who were attacking nodes because they don't like what some other hackers are able to do there, you know, like competing something as a service, you know, fill in the blank. You know, Malware as a Service, Hacking as a Service, you know, anything as a service stuff. So, fine.

When I caught this recent news of a just-published survey, it helped to resolve the mystery surrounding why was it that so many school districts were failing and falling to ransomware attacks. Get a load of this. According to a study published by the Consortium for School Networking, a professional association of school systems technology providers, fully two out of every three school districts - and this is districts, not just individual schools. Two out of three districts do not employ the full-time services of someone specializing in cybersecurity. Districts. And one in eight districts don't even allocate any funds for cybersecurity defense whatsoever. So to me, in this day and age, that's unconscionable and astonishing.

Think of the challenge that they're facing. Large enterprises have it tough enough where they're servicing employees on their networks who are at least trying to do the right thing by not clicking on everything they receive in email. But a school district of any appreciable size would be an insanely complex network to secure, especially given that the interior of its network is filled to the brim with rambunctious children and teenagers, half of whom are probably attempting to hack their school's network from the inside. I just can't imagine how you secure a school district's network.

And as an aside I'll just note that it's a VERY GOOD THING, in bold capitals, that the Internet didn't happen while I, well, was there, that it didn't happen until after I'd already attended my high school's 10th reunion. As it was, remember, I told the story of the "Portable Dog Killer" adventure that had the district's technicians climbing around in the rafters trying to locate the source of the near ultrasonic sounds that everyone was hearing that day.

And I mentioned at the end of the story that Vice Principal Archibald knew me on sight. Remember he faked me out by suddenly spinning around after he was deliberately walking away, and then he pointed at me. Now, I don't recall whether I mentioned that one of the many reasons that Vice Principal Archibald knew me by name was because at one point I was caught holding a copy of the master janitor's grand master key to the entire San Mateo Union High School district which unlocked every single school door district-wide.

Leo: Oh, boy.

Steve: So, yeah, they knew my name. So I shudder to think what would have happened had I been there once the school was networked because, boy, that would have been - would've been a lot of fun.

Leo: Oh, man, you would have had a fun time.

Steve: Oh, yeah. Anyway, so the idea that today so little attention is being paid to cybersecurity in two out of every three school districts, well, that's just nuts. That's, I mean, I don't want to say they deserve what they're getting. Nobody deserves ransomware attacks and having all your servers scrambled and encrypted. But, boy, you know, really. Maybe the message is finally sinking in.

Okay, now, normally innocuous civilian security cameras can be trouble, it turns out, in times of conflict. The Ukrainian Security Service, the SSU - probably Security Service for Ukraine - has asked its citizens to please disconnect any security cameras they may have which are aimed at public spaces. The SSU says Russia is exploiting vulnerabilities in modern security cameras to access these cameras' feeds, and they have proof that

Russia is using these feeds to time their launching of missile attacks and to adjust attack targeting in real-time.

After the SSU sent SMS messages to all Ukrainian citizens carrying this request last week, several Russian military bloggers suggested that the agency may be trying to mask the movement of its own troops, that is, Ukrainian troops, leading up to its impending Russian counteroffensive. And there may be some truth to that, too. But in any event, wow, talk about unintended side effects of otherwise innocent cameras that may be looking out into public areas and may not be as secure as people would hope they were.

We've recently been looking at the NSO Group with their Pegasus smartphone malware and other malware-for-hire groups. One of the worries surrounding the availability of off-the-shelf spyware tooling is that those who could never manage to do this themselves now only need money to purchase what they want, thus empowering them. A similar set of services has been emerging over the past several years which, for lack of any better term, we might call "cyber mercenaries," you know, hackers-for-hire. Exactly three years ago, in June, Reuters published an exclusive piece of reporting titled: "Obscure Indian cyber firm spied on politicians, investors worldwide."

The first two lines of their full report state: "A little-known Indian IT firm offered its hacking services to help clients spy on more than 10,000 email accounts over a period of seven years. New Delhi-based BellTroX (B-e-l-l-T-r-o-X) InfoTech Services targeted government officials..."

Leo: That sounds made up. That sounds like a movie name. All right. All right.

Steve: BellTroX Infotech Services.

Leo: Infotech Services.

Steve: Well, it does sound like an Indian IT name, I think. They have fun names - "...targeted government officials in Europe, gambling tycoons in the Bahamas, and well-known investors in the United States, including private equity giant KKR and short seller Muddy Waters, according to three former employees, outside researchers, and a trail of online evidence."

And today, The New Yorker magazine has a wonderful profile of this same firm, BellTroX. It's actually - so it's Be-l-l-T-r-o-X. So I don't know. I would think you'd say BellTroX. But, you know, funky capitalization. And what the New Yorker describes as India's budding cyber mercenary market, which outsources hacker-for-hire services across the globe, with the Indian government's tacit acceptance.

Now, okay. This podcast could make a full meal out of this coverage, but it's time for us to get to this week's very interesting discussion of the backdoors that have been carelessly designed into many of today's most popular motherboards. So I've provided the links to both of these fascinating stories for anyone who wants to take the time to learn more. Suffice to say it is now possible for those without any cyber hacking skills to simply rent such cyber skills from mercenaries of any skill level required to obtain whatever cyber outcome might be needed, for a fee.

I have two quick Closing-the-Loop pieces from our listeners. James Brooks said: "While I agree that the 'Request OTR' is a noble idea" - that was the topic of last week's podcast,

the idea that the Brave browser has implemented, which would allow a website to proactively ask a browser to ask its user if they would like to have their use of the site kept private and flushed the moment they leave. Anyway, so he says: "It's a noble idea." He says: "I have concerns that sites looking to victimize my kids will include this header. All they need to do is prep the user for the prompt, and my kids are on a dangerous site with little chance of me having visibility into it."

And so I thought that was an interesting point. I guess I would say that relying on browser history is probably not strong protection either. And I would be using a DNS service that didn't resolve dangerous sites, you know, as a good means of protection. But I certainly do take the point that this could be used by other sites that want to victimize their users and not leave a trail of that having happened.

Mark Newton, he said: "What are some good ways to block TLDs such as .zip and .mov if people don't have the ability to block it at their firewall? I was thinking the hosts file and adding a 127.0.0.1 with a *.zip. Any thoughts?" Okay. Unfortunately, the Windows HOSTS file won't handle wildcards. Be nice if it did. It doesn't. I'd say that the best solution would be to use an external security-oriented service such as NextDNS. You can set up your configuration in, for example, NextDNS to block entire TLDs. And that's something that I know our listeners, because I have some other feedback, have already done. So I think that's what I would do. The idea of having a security-oriented DNS provider for many purposes, both the ones that James Brooks brings up and that Mark Newton was asking about, I think makes a lot of sense. What does not make a lot of sense, Leo, is Windows Platform Binary Table.

Leo: Even though it has a great name.

Steve: Even though it's so catchy, you know, you can't even say the abbreviation quickly, WPBT. It's too bad it's not BLT. That'd be good. Windows BLT.

Leo: Microsoft is terrible at naming things.

Steve: Platform Loader Table, that'd be good, Windows BLT. But no. Binary Loader Table. No.

Leo: No, no, no.

Steve: The biggest news of the week, that rocked the security world, was the revelation that the very popular motherboards made by Gigabyte - and I'm typing these show notes on one right now, sitting in front of a machine that's based on a Gigabyte - were found to be secretly downloading code that the motherboard would then cause Windows machines to execute. And what was extra disturbing was that the TCP connection over which this download took place was neither authenticated nor encrypted.

Leo: Yeesh.

Steve: Yeah. To everyone's shock and horror, and the source of a great many terrific headlines, this meant that it would be trivial for bad guys to intercept these communications to install their own rootkit malware. Doesn't sound good. But if my

demeanor makes you wonder whether you're detecting that I'm somewhat less scandalized by these revelations than the rest of the security press, you would be correct. And that's not because all of the above is not true. It is all true. It's because we're all completely fooling ourselves in the belief that Windows operating systems actually offer any true security in the first place.

Leo: That's an angle I didn't really think of. Motherboard security, who cares? The operating system's full of holes.

Steve: Yeah, I mean, we're just glad it boots. Windows has never, never been a secure operating system, and to meet the demands of the marketplace it never can be. This should have been obvious when Windows was first placed onto the Internet and openly published everyone's C: drive to the entire world. It sounds insane to say that now, but we know it happened. I was ridiculed certainly after the birth of this podcast for suggesting that the original Windows metafile format (WMF) deliberately supported the ability to execute native code that was carried in the metafile. I'm absolutely certain that it did, just as I'm sure that it once made sense to the developer who had added that escape function.

But years later, the world was a very different place, so the industry was horrified by that discovery and thought that the only possible way it could happen was by mistake. Mark Russinovich reverse engineered the Windows' metafile interpreter, as I had; and he said, "It sure does look deliberate." There was never any doubt. My point is, context matters, and the world is constantly changing.

Last week I said that it's not fun to use a truly secure operating system because you can't actually get any work done. First of all, it's not at all clear that we even know how to create a secure operating system because we haven't figured out how to create secure software. So we're resorting to erecting various types of fencing around our admittedly insecure attempts by using hardware-enforced protection levels and sandboxes and virtual machine containers and so on. So, you know, we're still stumbling forward.

Microsoft has recently made headlines by announcing that they're going to rewrite the Windows kernel in RUST for much better security. That's great. But are they also going to then prohibit the use of any third-party peripheral device drivers, all of which run alongside their shiny new RUST code in the kernel? If not, then any notion of true security is just marketing hype. That's not to suggest that rewriting the kernel in RUST will not be useful. If you plug a hole in a large block of Swiss cheese, you do at least have one fewer holes.

So I wanted to kick off our discussion today of the Windows Platform Binary Table with a wee bit of a reality check. Make no mistake: What Eclypsium discovered was not good. So another previously unsuspected hole in the Swiss cheese has been plugged. But no one should imagine that doing so meaningfully increases Windows' actual security. That said, we do need to keep trying. The quest for security will, I think, guarantee employment for anyone who is capable and competent in the field. Even if school districts are not hiring, everyone else is.

Okay. So what's this Windows Platform Binary Table? It's a facility which Microsoft first defined and implemented 11 years ago, back in 2012, and it was first supported in Windows 8. It defines a clear, clean, and well-documented means for the platform from which Windows is booted - our motherboards - to provide Windows with its own code, previously stored within its firmware, which Windows, ever since Windows 8, will look for and execute when it's present as part of the Windows boot process.

Now, if your first thought is that this also perfectly describes the operation of a motherboard-based rootkit, you would be correct in your thinking. Because it was foreseeable that advanced motherboards might need to have the capability to reach up into the operating system to take advantage of its rich array of advanced services and connectivity, like downloading and installing their own firmware interface drivers, or perhaps updating their own firmware itself, and since Microsoft did not want motherboards each inventing their own horrible kludges in order to do this, Microsoft formalized this capability in what's known as the Windows Platform Binary Table.

And since this podcast likes to stay on top of such things, I should note that this is not the first time we've talked about this here. Security Now! Episode 521 which we recorded on August 18, 2015, was titled "Security Is Difficult." Okay. I don't know if that was a new concept for us back then, but that's what we called that podcast. And during that podcast we discussed the Windows Platform Binary Table facility. This was in the context at the time of Lenovo laptops which were found to be behaving badly.

And this is also not the first time that Eclypsium has surfaced with some worrisome news regarding the operation of Windows WPBT. Eclypsium's posting of September 23rd, 2021, so about a year and a half ago, not quite, was titled "Everyone Gets a Rootkit." I'll share their Executive Summary from that posting since it serves to set the stage for what follows today.

Not quite two years ago, they wrote: "In a connected, digitally transformed age, the term 'no good deed goes unpunished' could perhaps be rephrased as 'no feature goes unexploited.'" Or as we like to say here on this podcast, what could possibly go wrong? "The protocol, called Advanced Configuration and Power Interface (ACPI) was introduced in the early 2000s when it became apparent that the energy consumption of billions of rapidly proliferating computing devices was a significant and increasing drain on national and regional energy supplies."

I remember that. It was like this was - remember you could buy green computers. That was a big thing. Everyone was like, green monitors and green computers. It's like, okay. "So ACPI was designed to efficiently manage energy consumption on PCs, along with several additional well-meaning use cases. As laptop usage and portable computing became universal demands, ACPI became a de facto standard for nearly all systems." And yes, I think it's on everything now.

"With the advent," they wrote, "of Windows 8, the protocol evolved to include an object called the Windows Platform Binary Table and has since been included in every single Windows OS shipped since 2012. In June of 2021, so what's that, that's exactly two years ago, Eclypsium researchers discovered significant flaws in WPBT. These flaws make every Windows system vulnerable to easily crafted attacks that install fraudulent vendor-specific tables. These tables can be exploited by attackers with direct physical access, with remote access, or through manufacturer supply chains. More importantly, these motherboard-level flaws can obviate initiatives like Secured-Core and Secure Boot because of the ubiquitous usage of ACPI and WPBT. Security professionals need to identify, verify, and fortify the firmware used in their Windows system."

Okay. So that was their overview. A bit later, here's their description of the issue at the core of this problem: "The Eclypsium research team" - this is, again, two years ago, two and a half - "has identified a weakness in Microsoft's WPBT capability that can allow an attacker to run malicious code with kernel privileges when a device boots up. WPBT is a feature that allows OEMs to modify the host operating system during boot to include vendor-specific drivers, applications, and content. Compromising this process can enable an attacker to install a rootkit compromising the integrity of the device.

"The issue stems from the fact that, while Microsoft requires a WPBT binary to be signed, it will accept an expired or revoked certificate. This means an attacker can sign a malicious binary with any readily available expired certificate. This issue affects all Windows-based devices going back to Windows 8 when WPBT was first introduced. We've successfully demonstrated the attack on modern, Secured-Core PCs that are running the latest boot protections. This weakness can be potentially exploited via multiple vectors, for example, physical access, remote, and supply chain; and by multiple techniques, malicious bootloader, DMA, et cetera. Organizations will need to consider these vectors and employ a layered approach to security to ensure that all available fixes are applied, and identify any potential compromises to devices."

And then they quote Microsoft: "Microsoft recommends customers use Windows Defender Application Control (WDAC) to limit what is allowed to run on their devices. WDAC policy is also enforced for binaries included in the WPBT and should mitigate this issue. We recommend customers implement a WDAC policy that is as restrictive as practical for their environment." Yeah, it's easy for Microsoft to say. It's difficult to live with, as I said. If you have a really secure computer, you can't get any work done. Then they said: "You can find documentation on WDAC," and then we provide a link.

So, okay. It makes sense that WPBT-based code execution would have these problems. First, all certificates expire, and all code-signing certificates expire. Right now, at this moment, my latest code-signing certificate from DigiCert has expired. I haven't renewed it since I've been working on SpinRite in DOS, so there's been no need. But the moment I get the DOS side finished, I'll be integrating it into its Windows executable.

So the first thing I'll do is renew that cert. But right now it's expired. And the Windows executables it signed - SQRL, InControl, and the DNS Benchmark - all run right now without complaint and show that the signature is valid. The reason for this is that, unlike for HTTPS TLS connections which, you know, unlike those for code signing, the only requirement is that the certificate be valid at the time of that signing. That's the test that's employed. So it's unsurprising that Eclypsium discovered this, although it is a bit distressing. And it turns out that's not even a requirement for signing code that is loaded through WPBT.

The second issue, of continuing to honor proactively revoked code-signing certificates is also worrisome. As we know, when valid certificates are found to have escaped into the wild, their revocation until they naturally expire on their own, which wouldn't help us either, is our only recourse. What Eclypsium found was that Windows is not checking for certificate revocation at this early time during Window's booting. Perhaps it's unable to do so. So this means that once someone obtains a code-signing certificate - which is not a high bar to jump over if you want to actually get your own, and which has a three-year lifetime - even if that certificate is discovered being misused in the wild, and it's revoked, Windows will continue to honor any boot time code that was signed by it.

And the situation might even be worse, since it doesn't appear that even a certificate's validity, as I mentioned, at the time of signing is required. Here's what they said. They said, and this is all still two and a half years ago: "WPBT requires any binaries to be properly signed. Microsoft's WPBT code-signing policy states" - this is Microsoft - "All binaries published to Windows using the WPBT mechanism outlined in this paper must be embedded signed and timestamped. These images should be linked with the /INTEGRITYCHECK option and signed using the SignTool command with the /nph switch to suppress page hashes."

Then Eclypsium said okay, that's all well and good, that's what they're saying. "However," they said, "our testing revealed that this check will pass even if the code-signing certificate has been explicitly revoked. To highlight this, we signed our malicious test code using a Hacking Team code-signing certificate that was revoked in 2015."

Okay, and they did this in 2021 after that cert would have expired. So not only revoked, but expired.

They said: "This and many other expired or revoked certificates are readily available to anyone on GitHub. This particular cert was revoked when the company's tools including a UEFI rootkit which was exposed in a major breach. We used this certificate just to highlight a particularly egregious example, but the same process would work with other revoked certificates, as well. This attack is also significant because it allows an attacker to drop a malicious file in the OS, even while BitLocker encrypts the disk. BitLocker would prevent other types of threats such as the Hacking Team implant, which used firmware to directly write the malicious code to disk. However, in the case of WPBT, an attacker could avoid BitLocker since the malicious file is pushed to the OS and stored in c:\windows\system32 and run on startup, regardless of BitLocker."

Okay. So in other words, this entire WPBT mess is a huge hole in Windows boot security. Like I said, security? You know, marketing hype. Windows is not secure and is arguably not securable. Essentially, motherboards must be absolutely and utterly trusted because they, with Microsoft's full blessing, and a spec for how to do this, are able to completely subvert the security of the Windows boot process.

And as we also saw, Microsoft's only solution and recommendation is for those concerned by this behavior to manually apply their Windows Defender Application Control, which, for what it's worth, is active at this early boot stage. So it can be used to govern what the motherboard is able to inject and then cause to run. Unfortunately, WDAC is typically used in a blacklisting mode, not in a whitelisting mode, because whitelisting isn't at all practical in the real world, just like true security isn't practical. This means that the specific filename being used and injected into the system32 directory would need to be known, and you would need to know that the firmware was never going to change that name to something different, and how would you know?

So with this background two and a half years ago, we're caught up with today, which makes the conclusion I previously drew all the more chilling. I said: "Motherboards must be absolutely and utterly trusted because they are able to completely subvert the security of the Windows boot process."

What put Eclypsium back in the news Wednesday of last week and generated so many hysterical headlines was that they discovered that Gigabyte motherboards were not injecting and running implant code into Windows - no, wait, I've got a "not" in there that shouldn't be. They discovered that Gigabyte motherboards, but on the other hand all other motherboards are, too, I mean, all manufacturers are doing this now. They discovered Gigabyte motherboards were injecting and running implant code into Windows, and that this code was then downloading and running code from non-secured web servers over HTTP, which, you know, that means they could be in Russia, right, where they use HTTP; or unenforced HTTPS.

Here's what Gigabyte posted. They said: "Recently, the Eclypsium platform began detecting suspected backdoor-like behavior within Gigabyte systems in the wild. These detections were driven by heuristic detection methods, which play an important role in detecting new, previously unknown supply chain threats, where legitimate third-party technology products or updates have been compromised.

"Our follow-up analysis discovered that firmware in Gigabyte systems is dropping and executing a Windows native executable during the system startup process, and this executable then downloads and executes additional payloads insecurely. It uses the same techniques as other OEM backdoor-like features like Computrace, a.k.a. LoJack DoubleAgent, which is abused by threat actors and even firmware implants such as Sednit LoJax, MosaicRegressor, and VectorEDK. Subsequent analysis showed that this

same code is present in hundreds of models of Gigabyte PCs. We are working with Gigabyte to address this insecure implementation of their app center capability."

They wrote: "In the interest of protecting organizations from malicious actors, we are also publicly disclosing this information and defensive strategies on a more accelerated timeline than a typical vulnerability disclosure." In other words, this is so bad it can't wait. "This backdoor appears to be implementing intentional functionality and would require a firmware update to completely remove it from affected systems." In other words, it's not a bug, it's a feature.

"While our ongoing investigation has not confirmed exploitation by a specific threat actor, an active widespread backdoor that is difficult to remove poses a supply chain risk for organizations with Gigabyte systems. At a high level, the relevant attack vectors include compromise in the supply chain, compromise in the local environment, or malware persistence via functionality of this firmware in the systems. A more detailed analysis of these risks is provided with suggested mitigations. After a more traditional vulnerability disclosure timeline, we plan to publish details about how this works." Which will be, you know, why? We all know how it works.

"There are two important aspects of our findings. First, Eclypsium automated heuristics detected firmware on Gigabyte systems that drops an executable Windows binary that is executed during the Windows startup process. And second, this executable binary insecurely downloads and executes additional payloads from the Internet. So that's the key issue here. Without any ability to intervene, observe, or control, hundreds of Gigabyte motherboard models have the ability [and are] causing Windows to go out onto the Internet to fetch and run additional Windows executables every time a system is booted. And what's worse, this is done by fetching those files over unauthenticated and unencrypted HTTP."

So in their analysis of the firmware which is dropping the OS executable they said: "An initial analysis of the affected UEFI firmware identified the following file." And I can't even read this. It starts at 8ccbee6f7858ac6b9, and that's about the first quarter of it. And it ends in 0c715971b2.bin. So that's the file, a .bin file with a ridiculous name.

They say: "This is a Windows Native Binary executable embedded inside of UEFI firmware binary in a UEFI firmware volume. This Windows executable is embedded into UEFI firmware and written to disk by firmware as part of the system boot process, a technique commonly used by UEFI implants and backdoors." In other words, it's doing what bad things do. They said: "During the Driver Execution Environment (DXE) phase of the UEFI firmware boot process, the WPBTDXE.efi firmware module loads the embedded Windows executable into memory, installing it into a WPBT" - that's the Windows Platform Boot Table - "ACPI table which will later be loaded and executed by the Windows Session Manager Subsystem." That's smss.exe. That's the thing that actually does this.

And they go on to explain that although this setting appears to be - oh. This is important. I shouldn't skip this. The WPBTDXE.efi module checks if the APP Center Download and Install feature has been enabled in the BIOS/UEFI setup before installing the executable into the WPBT ACPI table. They said: "Although this setting appears to be disabled by default, it was enabled on the system we examined." So that's first thing to know.

This behavior is something that appears to be configurable by the user in the firmware setup. And they're saying it was off by default, which is fabulously good news. APP Center Download and Install feature was on on the one that was generating this traffic, they say disabled by default. So that's good news. I guess the only danger would be if someone's setting it up, or maybe a consultant who sets it up and wants this to be all taken care of, thinks that this is a good thing to turn on, so it may be enabled. You'll have to see.

This executable uses the Windows Native API to write the contents of an embedded executable to the file system at, you know, the system root, typically c:\windows, then \system32\, and it's called GigabyteUpdateService.exe. It then sets registry entries to run this executable as a Windows Service. So it's creating a service which Windows will be running. The mechanism described here is similar to the methods used by other UEFI firmware implants such as LoJax, MosaicRegressor, MoonBounce, and VectorEDK, referenced previously.

Also note that this is not then transient. It does it presumably every time, but this Gigabyte update service, it's sticking around. It's physically written into the system32 directory, and it's defined as a service that will be started every time the system boots, whether or not you have subsequently disabled this APP Center Download and Install. So that's important. If you found it on, and you turned it off, you're not out of the woods because it being on previously meant that GigabyteUpdateService.exe was written to system32 and defined as a service.

Okay. So then secondly they talk about downloading and running further executables. The dropped Windows executable, that's that Gigabyte update service, is a .NET application. It downloads and runs an executable payload from one of the following locations, depending upon how it's been configured. And they give us three different URLs.

The first one is http://, and it starts with mb.download.gigabyte.com/, then FileList/, and then Swhttp and then /LiveUpdate4 is the last directory. The second URL is an HTTPS URL, so that's interesting. And otherwise the URL is identical. So that says they're running servers on both port 80 and on port 443 and accepting connections either insecurely on HTTP or over TLS with HTTPS. And the third one is also interesting. It's not a fully resolved public domain. It's https://software-nas/Swhttp/LiveUpdate4. So that would be a locally resolved IP that would allow Gigabyte motherboards to go download stuff inside of an internal LAN, not reaching out into the public.

They said: "Plain HTTP, the first bullet" - the first URL - "should never be used for updating privileged code [duh] as it is easily compromised via machine-in-the-middle attacks. However, we noticed that even when using the HTTPS-enabled options, remote server certificate validation is not implemented correctly. Therefore, MITM (man-in-the-middle) is possible in those cases also." In other words, insecurely.

"The firmware does not implement any cryptographic digital signature verification or any other validation over the executables it downloads. The dropped executable and the normally-downloaded Gigabyte tools do have a Gigabyte cryptographic signature that satisfies the code-signing requirements of Microsoft Windows, but this does little to offset malicious use, especially if exploited using Live-off-the-Land techniques like in the recent alert regarding Volt Typhoon attackers. As a result, any threat actor can use this to persistently infect vulnerable systems either via MITM or compromised infrastructure.

"These issues expose organizations to a wide range of risks and attack scenarios." And we've got four bullets. "First, abuse of an OEM backdoor by threat actors. Previously," they say, "threat actors have taken advantage of legitimate but insecure/vulnerable OEM backdoor software built into the firmware of PCs. Most notably, Sednit group, which is APT28, also known as Fancy Bear, exploited Computrace LoJack to masquerade as legitimate laptop antitheft features.

"Secondly, compromise of the OEM update infrastructure and supply chain. Gigabyte does have documentation on their website for this feature, so it may be legitimate. But we cannot confirm what is happening within Gigabyte. In August of 2021 Gigabyte experienced a breach of critical data by the RansomExx group, and then experienced another breach in October of 2021 by the AvosLocker group." In other words, their

security's not that great, apparently. And all these Gigabyte motherboards that have this thing enabled are downloading whatever happens to be on the server at the time.

They say: "Third, persistence using UEFI Rootkits and Implants is another vector. Fourth, man-in-the-middle attacks on firmware and software update features. And finally, ongoing risk due to unwanted behavior within official firmware. Backdoors hidden within UEFI or other firmware can be hard to remove. Even if the backdoor executable is removed, the firmware will simply drop it again the next time the system boots up. This challenge was demonstrated before when trying to remove Computrace LoJack and related to vulnerabilities in Lenovo Service Engine on notebooks and laptops."

Okay. So that's what was discovered, that Gigabyte motherboards were insecure in their downloading of whatever happened to be available on those URLs, and that Windows would then run those things.

The following day, immediately, on last Thursday June 1st, Gigabyte responded with their posting "Gigabyte Fortifies System Security with Latest BIOS Updates and Enhanced Verification." Their posting is short, so I'll share it. They said: "June 1st, 2023. Gigabyte Technology, one of the leading global manufacturers of motherboards, graphics cards, and hardware solutions, has always prioritized cybersecurity and information security." Okay. All evidence to the contrary. "Gigabyte remains committed to fostering close collaboration with relevant units and implementing robust security measures to safeguard its users.

"Gigabyte engineers have already mitigated potential risks and uploaded the Intel 700/600 and AMD 500/400 series Beta BIOS to the official website after conducting thorough testing and validation of the new BIOS on Gigabyte motherboards. To fortify system security, Gigabyte has implemented stricter security checks" - or you might say any security checks - "during the operating system boot process. These measures are designed to detect and prevent any possible malicious activities, providing users with enhanced protection.

"First, Signature Verification: Gigabyte has bolstered the validation process for files downloaded from remote servers. This enhanced verification ensures the integrity and legitimacy of the contents, thwarting any attempts by attackers to insert malicious code.

"Two, Privilege Access Limitations: Gigabyte has enabled standard cryptographic verification of remote server certificates." What a concept. "This guarantees that files are exclusively downloaded from servers with valid and trusted certificates, ensuring an added layer of protection. BIOS updates for the Intel 500/400 and AMD 600 series chipset motherboards will also be released on the Gigabyte official website later today, along with updates for previously released motherboards. Gigabyte recommends that users regularly visit the official Gigabyte website for future BIOS updates." So that's great. If you have a Gigabyte motherboard, now would be a good time to update the BIOS. There are very insecure practices still in firmware until the BIOS is updated.

So we have some takeaways. If you have a Gigabyte motherboard, and you dislike the danger that's inherently presented by having it reaching out to anyone other than Microsoft to obtain unmanaged updates to your system, you'll likely want to disable APP Center Download and Install if it's present in your motherboard firmware. And as I noted, you'll also want to look in the system32 directory for the Gigabyte updater which has been installed as a service, and shut that down as a service. You could look in Windows Service Manager for it and stop it and then set it to disabled, and that would keep it from running.

And finally, the best news from all of this is there is a fully generic undocumented registry entry that can be added to Windows, any Windows 8 through presumably ever,

to shut down all of this Windows boot behavior. The registry key, it's under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\ Control\Session Manager, because remember it's the Session Manager because remember it's the session manager that's responsible for this behavior. And if you set - and I've got this all documented - "DisableWpbtExecution," you create a DWORD with that name and set it to one, Windows will no longer do this. No matter what motherboard you are running under, this will no longer happen.

The Windows Session Manager component looks for this registry key and will abort the execution of anything being provided by the motherboard if this setting is present. So that's the number one most robust thing that any Windows user can do to shut down and disable this almost certainly unwanted Windows behavior. Because it's universal and safe, I've made this registry file this week's GRC Shortcut of the Week. So just put "grc.sc/926" into the URL of any browser. It will obtain a zip file named DisableWPBT.zip, whose contents is exactly what's shown above, a .reg file which, when double-clicked, and then you confirm that it's what you want to do, will add that DisableWpbtExecution DWORD into the local machine's registry, giving it a value of one.

The reason that adding this disablement key to your Windows Registry is probably the right thing to do is that this Gigabyte event should serve us as a wakeup call. While this focused upon just Gigabyte, as I said before, all systems are now known to be using this mechanism for maintaining their firmware. Microsoft officially created this and blessed it and said this is what you should do.

And as for what could possibly go wrong, just ask those tens of thousands of HP OfficeJet 9020e printer users whether they wish they had not had their printer connected to the Internet early last month. In other words, supply chain problems can and do occur. As Eclypsium noted, Gigabyte has been previously penetrated twice. So having every Windows system happily downloading third-party software from who knows where, you know, seems like an unwarranted and unnecessary risk. Grc.sc/926. That will get you the zip file containing the reg file that you can apply to your Windows instance to stop it from doing that. And as we say, Leo, that's the show.

Leo: I'm glad you covered this. Yeah, we've been talking about it a little bit, as well. Of course the fact that they offered a patch firmware update is great. But a lot of people will never update their firmware. A lot of people don't listen to this show. And there's a problem. Therein lies the problem; right?

Steve: Yeah, yeah, big problem.

Leo: Yeah. That's why you listen to the show, and you should keep listening to the show every Tuesday.