Security Now! #926 - 06-06-23
Windows Platform Binary Table

This week on Security Now!

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?

"STRICTLY NO ACCESS" (No, really... Strictly None!)

Security News

Another week of silence from HP.

I started off this week looking to see whether there was any good news for the tens of thousands of owners of HP OfficeJet 9020e series printers which were all bricked that Monday morning, four weeks and a day ago. And again, total silence from HP. Against that backdrop, the first two headlines that popped up in my search for any news were stories published when this was news: *"HP rushes to fix bricked printers after faulty firmware update"* and *"HP Races to Fix Faulty Firmware Update That Bricked Printers."* So we have "rushes" and "races." 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 — after even more than a month — as been a big zero.

Mandatory "SMB Signing" coming to Windows 11

Microsoft has announced that future versions of Windows 11 will require all SMB messages to be cryptographically signed, regardless of the message type. That's a rather significant change and it's going to be interesting to see what it breaks. *"SMB"* stands for *"Server Messages Blocks"* and, among other things, it's what Windows File and Printer Sharing runs on top of. 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 troubled protocol.

Just over a year ago, Microsoft announced their related intention to remove all SMBv1 support from Windows 11. At that time, Ned Pyle, the Microsoftie making this announcement into Microsoft's TechNet Community: *"I had to save this behavior for last, it's going to cause a 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 can't connect to their old networked hard drive. I'll spread this 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."*

There are currently three major versions of SMB, 1, 2 and 3. 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 client and server roles has remained present. Remember that SMBv1 was first present in the MS-DOS Windows for Workgroups add-on which allowed DOS machines to participate as clients and servers on a Windows LAN. Since my 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 version 1 could be enabled under Windows 10. That's what I've been using to network my DOS machines. It has 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 contained SpinRite's source and executables for native source level debugging under DOS.

So last year Microsoft took this further by removing the binaries for SMBv1 from at least some of Windows 11. Apparently this still breaks things, so users of the higher-end editions are able to add it back if needed. But 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. That makes me suspect that this signing requirement is also likely to be somewhat soft and overridable.

Last Friday, the same guy, Ned Pyle, wrote: *"Heya folks, Ned here again. Beginning in Windows 11 Insider Preview Build 25381 (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."*

SMB signing is a simple but useful security mechanism that's been around since Windows 98 and Windows 2000, but it's never been forcibly enabled by default because of the signing overhead it adds to every message. Without signing, nothing detects or prevents the alteration or spoofing of SMB messages. Signing protects against NTLM relay attacks which have been a constant thorn in Microsoft's SMB implementation — so it's 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 may break some of the things that our listeners are using – things that either don't currently have SMB signing enabled or don't support SMG message signing.

OWASP

OWASP is the "Open Web Application Security Project" which has, for the past 20 years, since 2003, maintained the often quoted "OWASP Top Ten" 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.

We're talking about this today because OWASP has announced their work on a forthcoming "OWASP Top 10" for Large Language Model Applications. They wrote:

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 (LLMs). The project provides a list of the top 10 most critical vulnerabilities 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, suggest remediation strategies, and ultimately improve the security posture of LLM applications.

The following is a draft list of important vulnerability types for Artificial Intelligence (AI) applications built on Large Language Models (LLMs)

1. ***Prompt Injections***
   Prompt Injections bypass filters or manipulate the LLM using carefully crafted prompts that make the model ignore previous instructions or perform unintended actions.

2. ***Data Leakage***
   Accidentally revealing sensitive information, proprietary algorithms, or other confidential details through the LLM's responses.

3. ***Inadequate Sandboxing***
   Failing to properly isolate LLMs when they have access to external resources or sensitive systems, allowing for potential exploitation and unauthorized access.

4. ***Unauthorized Code Execution***
   Exploiting LLMs to execute malicious code, commands, or actions on the underlying system through natural language prompts.

5. ***SSRF Vulnerabilities***
   Exploiting LLMs to perform unintended requests or access restricted resources, such as internal services, APIs, or data stores.

6. ***Over reliance on LLM-generated Content***
   Excessive dependence on LLM-generated content without human oversight can result in harmful consequences.

7. ***Inadequate AI Alignment***
   Failing to ensure that the LLM's objectives and behavior align with the intended use case, leading to undesired consequences or vulnerabilities.

8. ***Insufficient Access Controls***
   Not properly implementing access controls or authentication, allowing unauthorized users to interact with the LLM and potentially exploit vulnerabilities.

9. ***Improper Error Handling***
   Exposing error messages or debugging information that could reveal sensitive information, system details, or potential attack vectors.

10. ***Training Data Poisoning***
    Maliciously manipulating training data or fine-tuning procedures to introduce vulnerabilities or backdoors into the LLM.

Some of these are old and generic, like Unauthorized Code Execution – that's not something anyone is going to want, except for the bad guys. Nor is Data Leakage, Inadequate Sandboxing, Server Side Request Forgeries, Insufficient Access Controls or Improper Error handling. 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 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 never quite disappear.

And then have a handful of new problems that are quite specific to this new application class.

“Prompt Injection” that’s a new one. So is “Over reliance on LLM-generated Content”, “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 Ten has, for 20 years, been a useful benchmark against which many projects have been measured. If nothing else, coders of all stripe were able to use it to double-check that they hadn’t overlooked something obvious, and yet easy to overlook. I expect this LLM-specific Top Ten list will serve a similar role. Someone should create one. QWASP is at work doing so.

Did Apple help the NSA attack the Kremlin?

http://www.fsb.ru/fsb/press/message/single.htm%21id%3D10439739%40fsbMessage.html

In the show notes, this next bit of news led with the question: *“Did **Apple** help the **NSA** attack the **Kremlin**?”* and I should have given it the subheading: *“Why does anyone take anything Russia says with even a grain of salt?”* Since I was unable to read the original, and doubtless comical, article in Russian, I'll rely upon the translation of the news which was published in the Risky Business newsletter. It said that:

Russia's FSB intelligence service claims to have uncovered a US 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 US National Security Agency (NSA), and claimed Apple cooperated with the NSA for the attacks. [THAT allegation was what caught my attention since, well, that would be HUGE if it turned out to be true, which seems unlikely in the extreme given everything we know about Apple and given the country that we, fortunately, still live in.]

*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 platform.”* [Advanced Persistent Threat] *Unlike the FSB, Kaspersky did not link the activity to the NSA or any other APT group and couldn't 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 1, 2023. [Which news we covered here at the time.] *Employees 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."* [Which no evidence supports.] *Russian officials asked the Prosecutor General's Office to start a formal investigation into Apple employees and US intelligence officials.*

And, unsurprisingly, in an email that the Risky Business newsletter received after its initial publication, 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."* Given the entire history of Apple's actions and the design of their devices which we have often examined closely, I certainly believe Apple's assertion far more than the Kremlin and the FSB who we catch frequently spewing State-supporting propaganda.

Kaspersky's analysis of this iPhone attack and compromise

Now, that said, if Kaspersky first saw this attack as early as 2019, assuming that the same 0-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 ago. I was interested in additional details (and something that an actual security firm created), so I tracked down Kaspersky's thoughts. They wrote:

While monitoring the network traffic of our own corporate Wi-Fi network dedicated for mobile devices using the Kaspersky Unified Monitoring and Analysis Platform (KUMA), 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 Toolkit's ***mvt-ios*** *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: triangulation[at]kaspersky.com.

Mobile device backups 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:

1. *The target iOS device receives a message via the iMessage service, with an attachment containing an exploit.*

2. *Without any user interaction, the message triggers a vulnerability that leads to code execution.*

3. *The code within the exploit downloads several subsequent stages from the C&C server that*

include additional exploits for privilege escalation.

4. *After successful exploitation, a final payload – a fully-featured APT platform – is downloaded from the C&C server.*

5. *Both the initial message and the exploit in the attachment is deleted.*

The malicious toolset does not support persistence, most likely due to the limitations of the OS. The timelines of multiple devices indicate that they may be reinfected after rebooting. The oldest traces of infection that we discovered happened in 2019. As of the time of writing in June 2023, the attack is ongoing, and the most recent version of the devices successfully targeted is iOS 15.7.

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 C&C server.

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 device's 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 will ever see is pseudo-random 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+ 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.

As we know, the Pegasus exploits are left behind in their target's 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 for the past four years... with no idea by whom or to what end. And no one, Kaspersky or anyone else, can go no 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 from discovery and elimination."*

The Trifecta Jackpot!
Last Thursday, Google announced what they called their *"Chrome Browser Full Chain Exploit Bonus"* program. Here's what they explained:

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 (last Thursday, which was June 1st) and until December 1st, 2023, 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. Your full chain exploit could result in a reward up to **$180,000** (and potentially more with other bonuses).*

*Any subsequent full chains submitted during this time are eligible for double the full reward amount! **So, $120,000 each.***

[So they're creating an extra incentive race among bug hunters. Anything found before this coming December yields double the normal bounty and the first person to supply an unknown exploit is rewarded with trouble the normal payout.]

We have historically put a premium on reports with exploits – "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 a 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 broad-scale mitigation strategies.

The show notes have a link with participation details for anyone who might be interested:
https://security.googleblog.com/2023/06/announcing-chrome-browser-full-chain.html

Who wrote that?
https://www.wicker.senate.gov/2023/5/wicker-scott-lankford-introduce-bill-to-increase-transparency-better-protect-children-online

Exactly one week ago, last Tuesday three quite conservative Senators in the Republican party introduced their "Know Your App Act." Their announcement carries the headline: *"Wicker, Scott, Lankford Introduce Bill to Increase Transparency, Better Protect Children Online"*

The announcement began:

*WASHINGTON – U.S. Senators Roger Wicker, R-Miss., Tim Scott, R-S.C., and James Lankford, R-Okla., 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 of 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. Roger Wicker leads with this 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. The Know Your App Act would bring much-needed transparency to app stores, empowering Americans to safeguard their families from exploitation."

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 common-sense 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 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, but the least Big Tech can do is label where Americans' money is going when they download in the app store."*

Wait... aren't all iPhones and iPads and pretty much iAnything also made in China? As well as most of the guts of our cars? Which is why that COVID-related Chinese chip shortage messed up U.S. and foreign automobile production so badly and jacked up the cost of used cars that already had all their chips in a line? The legislative announcement continues with:

As of March 2023, 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.

Wow. 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.

Products are routinely marked with their country of origin. So that's not any big new deal. But here we're implicitly saying that any applications made in China are inherently dangerous for that reason, which really feels wrong to me. And it's not going to work, anyway. If someone wants Temu, TikTok, CapCut, or Shein, that's what they are 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 quite unfair. Perhaps I'll be proven wrong in time, but today this seems quite wrong. Increasing the tension and division between two world superpowers doesn't seem like a winning strategy for anyone.

Tor gets anti-DoS protection
https://forum.torproject.net/t/alpha-release-0-4-8-1-alpha/7816
The Tor Project is testing a new Denial of Service mitigation feature for the Tor network where servers will require their connecting clients to solve a "puzzle" to access its resources. It's basically a CAPTCHA. 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. 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.

Cybersecurity at Educational institutions
https://www.cosn.org/tools-and-resources/resource/2023-state-of-edtech-leadership-survey/
https://www.cybersecuritydive.com/news/one-third-districts-have-cybersecurity-employees/651578/
When I caught this recent news of a just-published survey, it helped to resolve the mystery surrounding why so many school districts were 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 three of school **districts** – districts, not just schools – do not employ the full-time services of someone specializing in cybersecurity. And one in eight districts also do not allocate **any** funds for cybersecurity defense whatsoever. To me, that's unconscionable and astonishing.

Think of the challenge. A large enterprise's network is servicing employees who are at least trying to do the right thing by not clicking on everything they receive. 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.

As an aside I'll note that it is a **very good thing** that the Internet didn't happen until I had already attended my high school's 10th reunion. As it was, 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 menated at the end of that story that Vice Principal Archibald knew me on sight when he faked me out by suddenly spinning around and pointing at me. I don't recall whether I mentioned that one of the many reasons he knew me was because at one point I was caught holding a copy of the master janitor's grand master key to the entire district which unlocked every school door district wide.

So, yeah, I shudder to think what would have happened had I been there once the school was networked. But the idea that even today so little attention is being paid to cybersecurity in two out of every three school districts. That's just nuts.

Civilian Surveillance Cameras in Ukraine
Normally innocuous civilian security cameras can be trouble in times of conflict. The Ukrainian Security Service, the SSU, 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 camera' 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 was trying to mask the movement of its troops leading up to its impending Russian counter-offensive. And there may be some truth to that, too. But in any event, wow... talk about unintended side effects!

Cyber Mercenaries
We've recently been looking at the NSO Group with Pegasys 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 and they can be empowered.

A similar set of services have been emerging over the past several years which, for lack of any better term, we might call "Cyber Mercenaries" or 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 InfoTech Services*** *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 and what they describe as India's budding cyber mercenary market, which outsources hacker-for-hire services across the globe, with the Indian government's tacit acceptance.

This podcast could make a full and very interesting 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 some time to learn more. Suffice to say that it is now possible for those without **any** cyber hacking skills to simply rent such cyber skills from mercenaries of any skill level to obtain whatever cyber outcome might be required, for a fee.

https://www.newyorker.com/news/annals-of-crime/a-confession-exposes-indias-secret-hacking-industry

https://www.reuters.com/article/us-india-cyber-mercenaries-exclusive/exclusive-obscure-indian-cyber-firm-spied-on-politicians-investors-worldwide-idUSKBN23G1GQ

Closing the Loop

James Brooks / @oran0007

While I agree that the "Request OTR" is a noble idea, I have concerns that sites looking to victimize my kids will include this header. All they need do is prep the user for the prompt and my kids are on a dangerous site with little chance of me having visibility of it.

Mark Newton / @Lmarkn

*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 127.0.0.1 *.zip, etc. Thoughts?*

Unfortunately, the Windows HOSTS file won't handle wildcards. I'd say that the best solution would be to use an external security oriented service such as NextDNS. You can setup your configuration to block entire TLDs.

Windows Platform Binary Table

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

Now, if, from my demeanor, 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 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. Windows has 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 shortly 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 by the metafile. I'm absolutely certain that it did, just as I'm sure that it once made sense to the developer who added that escape. 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 have happened was by mistake. Mark Russinovich reverse engineered the Windows' metafile interpreter, as I had, and 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 resorted 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. 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 3rd-party peripheral device drivers, all 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"** — or **"WPBT"** ??

It's a facility which Microsoft first defined and implemented eleven years ago in 2012 and first supported in Windows 8. It defines a clear, clean and well documented means for the platform from which Windows is booting – our motherboards – to provide Windows with its own code, previously stored within its firmware, which Windows, since Windows 8, will look for and execute when 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 installing their own firmware interface drivers, or perhaps updating their own firmware, and since Microsoft didn't want motherboards inventing 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". And during that podcast we discussed the Windows Platform Binary Table facility. This was in the context 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 of 2021 was titled: "Everyone Gets a Rootkit." I'll share their Executive Summary from that posting since it server 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 good feature goes unexploited". 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. ACPI was designed to efficiently manage energy consumption in 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.

With the advent of Windows 8, the protocol evolved to include an object called the Windows Platform Binary Table (WPBT) and has since been included in every single Windows OS shipped since 2012.

In June 2021, 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 because of the ubiquitous usage of ACPI and WPBT. Security professionals need to identify, verify and fortify the firmware used in their Windows systems.

And a bit later, here's their description of the core issue:

The Eclypsium research team 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 have 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 (e.g. physical access, remote, and supply chain) and by multiple techniques (e.g. malicious bootloader, DMA, etc). 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.

"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. You can find documentation on WDAC – https://docs.microsoft.com/windows/security/threat-protection/windows-defender-application-control/wdac-and-applocker-overview"

It makes sense that WPBT-based code execution would have these problems. First, all certificates expire. And all code signing certificates expire. Right now, 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 without complaint and show that the signature is valid. The reason for this is that unlike for HTTPS TLS connections, 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, though it is a bit distressing.

But the second issue, of continuing to honor proactively revoked code signing certificates is far more worrisome. As we know, when valid certificates are found to have escaped into the wild, their revocation until they naturally expire 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. So this means that once someone obtains a code signing certificate – which is not a high bar to jump over – and which has a 3 three year lifetime, even if that certificate is discovered being misused in the wild, 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 at the time of signing is required. Eclypsium wrote:

WPBT requires any binaries to be properly signed. Microsoft's WPBT code-signing policy states:

> 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-line tool with the /nph switch to suppress page hashes.

However, 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 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.] *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 were 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.

In other words, this entire WPBT mess is a huge hole in Windows' boot security. Essentially, motherboards must be absolutely and utterly trusted because they 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 (WDAC), 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 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.

So with this background, we're caught up with today which makes the conclusion I previously drew all the more chilling. Remember that 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, but that this code was then downloading and running code from non-secured web servers over HTTP or un-inforced HTTPS. Here's what Gigabyte posted:

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 backdoor (a.k.a. LoJack DoubleAgent) abused by threat actors and even firmware implants such as Sednit LoJax, MosaicRegressor, and Vector-EDK. 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.

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. This backdoor appears to be implementing intentional functionality and would require a firmware update to completely remove it from affected systems. 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*
- *Malware persistence via functionality of this firmware in 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.

There are two important aspects of our findings:

- *Eclypsium automated heuristics detected firmware on Gigabyte systems that drops an executable Windows binary that is executed during the Windows startup process.*

- *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 to cause 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 files over unauthenticated and unencrypted HTTP.]

Stage 1: Firmware dropping OS executable

An initial analysis of the affected UEFI firmware identified the following file:

8ccbee6f7858ac6b92ce23594c9e2563ebcef59414b5ac13ebebde0c715971b2.bin

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.

During the Driver Execution Environment (DXE) phase of the UEFI firmware boot process, the "WPBTDXE.efi" firmware module loads the embedded Windows executable file into memory, installing it into a WPBT ACPI table which will later be loaded and executed by the Windows Session Manager Subsystem (smss.exe) upon Windows startup. The "WPBTDXE.efi" module checks if the "APP Center Download & Install" feature has been enabled in the BIOS/UEFI Setup before installing the executable into the WPBT ACPI table. Although this setting appears to be disabled by default, it was enabled on the system we examined.

[So that's a bit of very good news. Unless a user thought that this might be a good thing to turn on, it may be that this entire facility is disabled by default.]

This executable uses the Windows Native API to write the contents of an embedded executable to the file system at the following location:

%SystemRoot%\system32\GigabyteUpdateService.exe

It then sets registry entries to run this executable as a Windows Service. The mechanism described here is similar to the methods used by other UEFI firmware implants such as LoJax, MosiacRegressor, MoonBounce, and Vector-EDK, referenced previously.

Stage 2: Downloading and running further executables

The dropped Windows executable is a .NET application. It downloads and runs an executable payload from one of the following locations, depending on how it's been configured:

- *http://mb.download.gigabyte.com/FileList/Swhttp/LiveUpdate4*
- *https://mb.download.gigabyte.com/FileList/Swhttp/LiveUpdate4*
- *https://software-nas/Swhttp/LiveUpdate4*

Plain HTTP (the first bullet above) should never be used for updating privileged code as it is easily compromised via Machine-in-the-middle (MITM) attacks. However, we noticed that even when using the HTTPS-enabled options, remote server certificate validation is not implemented correctly. Therefore, MITM is possible in those cases also.

The firmware does not implement any cryptographic digital signature verification or any other validation over the executables. 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 Living-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.

- *Abuse of an OEM backdoor by threat actors – Previously, threat actors have taken advantage of legitimate but insecure/vulnerable "OEM backdoor" software built into the firmware of PCs. Most notably, Sednit group (APT28, FancyBear) exploited Computrace LoJack to masquerade as legitimate laptop anti-theft feature.*

- *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 2021, Gigabyte experienced a* *breach of critical data by the RansomEXX group and then experienced another breach in October 2021 by the AvosLocker group.*

- *Persistence using UEFI Rootkits and Implants – UEFI rootkits and implants are some of the stealthiest and most powerful forms of malware in existence. They reside in firmware on motherboards or within EFI system partitions of storage media, and execute before the operating system, allowing them to completely subvert the OS and security controls running in higher layers. Additionally, since most of the UEFI code exists on the motherboard instead of storage drives, UEFI threats will easily persist even if drives are wiped and the OS is reinstalled. The rate of discovery of new UEFI rootkits has accelerated sharply in recent years as seen by the discovery of LoJax (2018), MosaicRegressor (2020), FinSpy (2021) ESPecter (2021), MoonBounce (2022), CosmicStrand (2022), and BlackLotus (2023). Most of these were used to enable persistence of other, OS-based malware. These Gigabyte firmware images and the persistently dropped Windows executable enable the same attack scenario. Often, the above implants made their native Windows executables look like legitimate update tools. In the case of MosaicRegressor, the Windows payload was named "IntelUpdater.exe"*

- *MITM attacks on firmware and software update features – Additionally, the insecure nature of the update process opens the door to MITM techniques via a compromised router, compromised device on the same network segment, DNS poisoning, or other network manipulation. It is also important to note that the third connection option, https://software-nas/Swhttp/LiveUpdate4 , is not a fully qualified domain name, but rather, a machine name that would presumably be on the local network. This means an attacker on a local subnet could trick the implant into connecting to their system, without the need for DNS spoofing.*

- *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 relates to vulnerabilities in Lenovo Service Engine on notebooks and desktops.*

Gigabyte responded immediately, the following day, last Thursday June 1st, with their posting *"GIGABYTE Fortifies System Security with Latest BIOS Updates and Enhanced Verification."*

June 1, 2023 – GIGABYTE Technology, one of the leading global manufacturers of motherboards, graphics cards, and hardware solutions, has always prioritized cybersecurity and information security. 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 during the operating system boot process. These measures are designed to detect and prevent any possible malicious activities, providing users with enhanced protection:

1. 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.*

2. Privilege Access Limitations: GIGABYTE has enabled standard cryptographic verification of remote server certificates. 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, our takeaways for this are:

If you have a Gigabyte motherboard and you dislike the danger that's inherently presented by having it reach out to anyone other than Microsoft to obtain unmanaged updates to your system, you'll likely want to disable "APP Center Download & Install" if it's present in your motherboard firmware.

Assuming that Gigabyte doesn't change the name of the executable that they inject into Windows' System32 directory, WDAC could be used to blacklist *"GigabyteUpdateService.exe"*.

And finally, there is a fully generic undocumented registry entry that can be added to shutdown all of this Windows behavior:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager]
"DisableWpbtExecution"=dword:00000001
```

The Windows Session Manager component looks for this registry key and will abort the execution of anything provided by the motherboard if this setting is present. So that's the #1 most robust thing that any Windows user can do to shut down and disable this likely-unwanted Windows behavior. Because it's universal and safe, I've made this registry file this week's GRC Shortcut of the week. So "grc.sc/926" will obtain a ZIP file named DisableWPBT.zip whose contents is exactly what's shown above, a .REG file which, when double-clicked and confirmed will add the "DisableWpbtExecution" DWORD to the local machine's registry.

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 as a wakeup call. While this focused upon Gigabyte, ALL systems are now known to be using this mechanism for maintaining their firmware.
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.

As Eclypsium noted, Gigabyte HAS been previously penetrated TWICE. So having every Windows system happily downloading third-party software from who knows where seemed like an unwarranted and unnecessary risk.

grc.sc/926