Security Now! #1023 - 04-29-25

Preventing Windows Sandbox Abuse

This week on Security Now!

• Why did a mysterious empty "inetpub" directory appear after April's Patch Tuesday? • And what new Windows Update crashing hack did this also create? • North Korea is now creating fake US companies to lure would-be employees. • The "Inception" attack subverts all GPT conversational AIs. • New information about data loss in unpowered SSD mass storage. • Lots of terrific feedback from our listeners. • How malware has taken to hiding inside the Windows Sandbox and what you can do to stop it.

User interface design is an art.

Security News

The mysterious "inetpub" mystery

I first noticed a mention of this in passing a week or two ago, but it wasn't until I focused upon catching up with all the recent news that I realized that this was something worth sharing here. And part of the reason for that is that even now, today, we still don't know what it's all about.

A few weeks ago I saw something about a folder named "inetpub." As someone who's been hosting websites using Microsoft's IIS – Internet Information Services – web server for decades, I'm as familiar with that directory name as I am with "Program Files" and other familiar Windows directory names. So I didn't think much about it. But whatever is going on has confused many people who've wondered why this mysterious and completely empty "inetpub" directory suddenly appeared on their Windows 11 machines after this month's April patch Tuesday? And, bizarrely, Microsoft says it's not a mistake, the empty directory must not be deleted, but won't explain exactly why.

There's been a lot of coverage of this in the tech press, but I'll share a lightly edited version of what Forbes' Davey Winder wrote about this recent mystery under the headline *"Microsoft's New Windows Update — 1 Billion Users Warned: Do Not Delete"*. Davey wrote:

The latest and somewhat confusing situation of Microsoft's making has come about as Windows users noticed a mysterious new folder after the most recent security update. A folder with no explanation and one which Microsoft has now warned a billion Windows users they must not delete.

As part of the April 8 Patch Tuesday security updates, Microsoft included a fix for CVE-2025-21204. This vulnerability in the critical Windows Update Stack, which is responsible for the management of Windows updates, no less, could lead to an attacker to elevate privileges locally. Something that the experts at SecurityVulnerability.io described as posing "a significant risk to organizations, as the compromised systems could allow attackers to execute unauthorized actions, potentially undermining the integrity and security of sensitive information and system operations."

I won't bore you with the technicalities of link resolution process manipulation that could enable hackers to access files and execute commands; just know it's pretty darn serious. SecuerityVulnerability.io wrote: "The ability to conduct unauthorized actions can severely impact the integrity of the affected systems resulting in potential disruptions of operations, implementation of malicious software, or further vulnerabilities being introduced into the network." Which is why Microsoft fixed it, and that's a good thing.

The way that Microsoft fixed it, however, is not so good. A lack of transparency is a particular bugbear of mine when it comes to anything security-related, and this vulnerability patch is no exception. The problem is that Microsoft created a new and empty folder with the security update, the appearance of which led to a totally understandable debate in tech forums and on Reddit as well as other social media platforms. What was this "inetpub" folder, how did it get there, is it dangerous, is Microsoft using it to collect data, and should I delete it?

According to a new Microsoft security advisory update, the answer to the last of these questions is a resounding no. Microsoft warned that Windows users must not delete the inetpub folder. Doing so would remove the vital security protections it provides, and the reason for it being created by this update in the first place.

An April 10 update to Microsoft's security advisory concerning CVE-2025-21204, entitled "Windows Process Activation Elevation of Privilege Vulnerability,"confirmed that "after installing the updates listed in the Security Updates table for your operating system, a new %systemdrive%\inetpub folder will be created on your device."

Microsoft went on to say that the folder installation was "part of changes that increase protection" but failed to explain precisely how. What I do know is that the inetpub folder itself usually comes as part of the Internet Information Services web server platform, enabled using Windows Features, but this update has created it whether the user has IIS installed or not.

And I'll just insert here that anyone who already **did** have IIS installed on their machine **will** definitely have that directory and **would** be expecting to have it. If you have the IIS service installed on your machine you cannot **not** have that directory — it's part of IIS. Davey continues:

More transparency is required, methinks, although not at the expense of tipping off potential attackers as to how the mitigation works, of course. I contacted Microsoft for a statement, but a spokesperson informed me that there was nothing else to add, other than the information contained within the security advisory, at this time. What I can say, however, is that as a security wonk, I strongly urge all Windows users to follow Microsoft's advice: "This folder should not be deleted regardless of whether Internet Information Services (IIS) is active on the target device."

All of which is OK, but what if you have already deleted the inetpub folder from your Windows installation? I mean, given the nature of the update and the social media conspiracy theories that surrounded it, I wouldn't be surprised if that were, indeed, the case for many users. I have already had a number of readers contact me to say they did just that and ask what they should do now. The answer is simple: restore it. The methodology required to do that is, thankfully, also pretty simple as long as you complete the six steps as follows:

1. *Head for the Windows Control Panel.*
2. *Click on Programs.*
3. *In the Programs & Features section, choose the Turn Windows features on and off option.*
4. *Scroll down through the Windows Features box that appears.*
5. *Tick the checkbox for Internet Information Services.*
6. *Click OK.*

Windows will then whirr and grind its cogs until the inetpub folder has been restored once more, and you can check your system drive to ensure that it is. By enabling IIS in this way the same folder is recreated as if Microsoft had dropped it there in a security update, and it will provide the same protections from Windows threats as well.

I looked elsewhere for additional clarification but everyone is telling the same story. The "WindowsLatest" site wrote:

Once IIS is installed, you don't need to make additional changes to Windows 11. Installing IIS will restore the folder. Microsoft told Windows Latest that users need to follow the IIS installation steps if they have accidentally deleted the folder. This empty folder must remain present on the Windows 11 system partition (%systemdrive%\inetpub) for the security patch

to function correctly. The folder provides "increased protection." According to Microsoft, turning on IIS creates the same folder with the same protection, and your PC will not be vulnerable.

And then in a later update to this article, Windows Latest added:

Update: Microsoft will not explain why the empty folder is required to apply the security fixes.

I'm annoyed by what strikes me as very lazy advice from Microsoft. Installing IIS onto a system is not a small thing, So it's ridiculous overkill to tell people to install IIS as a means to create a single empty directory. Presumably, the directory named "inetpub" requires specific user account privileges to be set on it. But given the power of Windows Powershell, I'm sure that a simple Powershell script could do exactly the same thing. So asking people to install a full web server just to create a directory is nuts.

But that said, randomly deleting directories that don't apparently serve any purpose is probably not a good idea, either. Power users who would tend to notice such things like to imagine that they are in charge of their Windows installation and environment, though this becomes less true with each iteration.

What I'm wondering is whether uninstalling IIS once it's been installed leaves that "inetpub" directory behind? If so, the second half of the lazy advice should be to then remove IIS after rebooting the system to first complete its installation and verify the existence of the "inetpub" directory. What's infuriating is that Microsoft won't tell us anything about why any of this is necessary... and while I'm sure they know what they're doing, the whole thing does seem somewhat bizarre. (Why didn't Microsoft just put a file in that directory with the name *"This directory created by Windows Update"* or perhaps just *"Do not delete this directory"* ?)

But wait! —> There's More!

Meanwhile, a prolific security researcher we frequently reference, Kevin Beaumont who used to tweat as @GossieTheDog, has posted into his blog on Medium under the headline: *"Microsoft's patch for CVE-2025–21204 symlink vulnerability introduces another symlink vulnerability."* Kevin explains:

Microsoft recently patched CVE-2025–21204, a vulnerability which allows users to abuse symlinks to elevate privileges using the Windows servicing stack and the c:\inetpub folder. To fix this, Microsoft precreates the c:\inetpub folder on all Windows systems from April 2025's Windows OS updates onwards. However, I've discovered this fix introduces a denial of service vulnerability in the Windows servicing stack that allows non-admin users to stop all future Windows security updates.

Non-admin (and admin) users can create junction points in c:

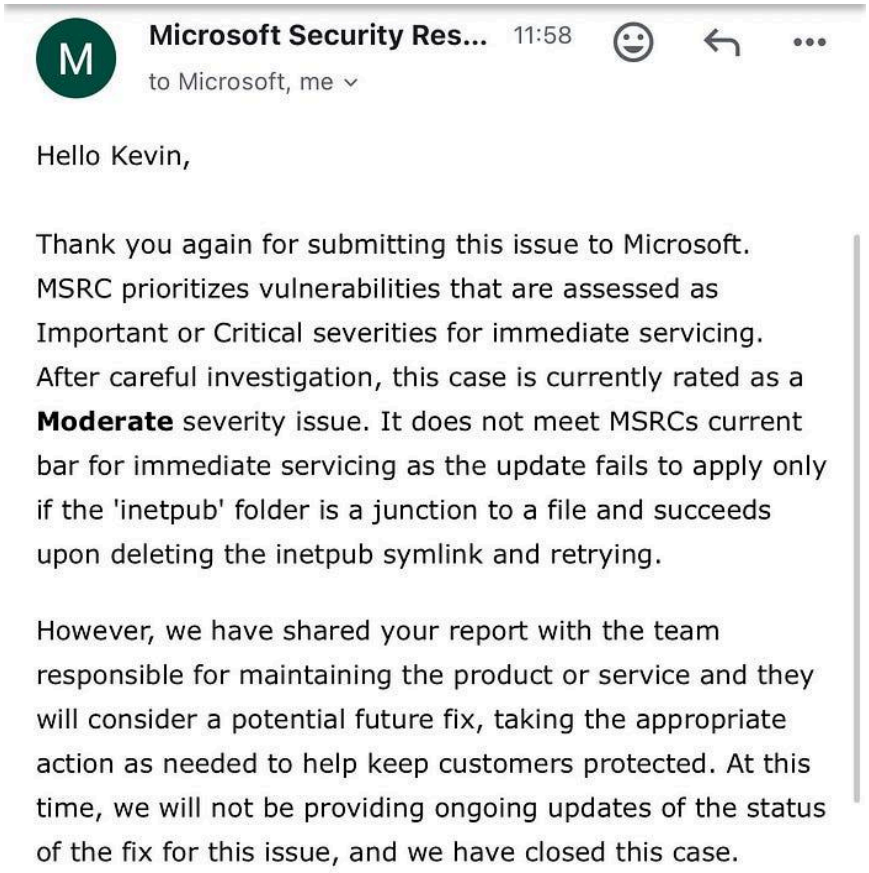
```
C:\Users\kevin>mklink /j c:\inetpub c:\windows\system32\notepad.exe
Junction created for c:\inetpub <<===>> c:\windows\system32\notepad.exe

C:\Users\kevin>
```

So a non-admin user can just do Windows+R, cmd, and then run:

C:\> mklink /j c:\inetpub c:\windows\system32\notepad.exe

This creates a symlink between c:\inetpub and notepad. After that point, April 2025 Windows OS update (and future updates, unless Microsoft fix it) fail to ever install — they error out and/or roll back, forcing the system to go without security updates.

I reported this to MSRC about two weeks ago, and finally received a response:



Microsoft Security Res... 11:58

to Microsoft, me

Hello Kevin,

Thank you again for submitting this issue to Microsoft. MSRC prioritizes vulnerabilities that are assessed as Important or Critical severities for immediate servicing. After careful investigation, this case is currently rated as a **Moderate** severity issue. It does not meet MSRCs current bar for immediate servicing as the update fails to apply only if the 'inetpub' folder is a junction to a file and succeeds upon deleting the inetpub symlink and retrying.

However, we have shared your report with the team responsible for maintaining the product or service and they will consider a potential future fix, taking the appropriate action as needed to help keep customers protected. At this time, we will not be providing ongoing updates of the status of the fix for this issue, and we have closed this case.

My feeling is the endpoint detection and response providers, including Microsoft, probably want to add detection for junction points being created from \inetpub on boot drives, as it looks like this issue isn't going to get patched any time soon, and it's a 100% reliable way to stop future security patching in Windows.

Whatever underlying problem Microsoft originally had with this CVE, it certainly feels as though someone cooked up a half-baked solution that wasn't very well thought out. The idea of needing to add an empty directory to the Windows file system which is normally only needed when a system is running their web server, and which is naturally then open to public abuse of the sort that Kevin stumbled upon seems really very, as I said, half-baked.

North Korea Strikes Again!

Okay, this one you're not going to believe! We've talked extensively about the challenge presented by employers who are attempting to do the right thing by not hiring spoofed employees from hostile foreign powers. Security researchers at the firm "Silent Push" just reported on their discovery of a new bizarre twist. Their headline was *"Companies to Deliver a Trio of Malware: BeaverTail, InvisibleFerret, and OtterCookie"* but that headline doesn't do the story justice. To give everyone a sense for what they discovered, they start with four key findings:

- *Silent Push Threat Analysts have uncovered three cryptocurrency companies that are actually fronts for the North Korean advanced persistent threat (APT) group Contagious Interview: BlockNovas LLC, Angeloper Agency, and SoftGlide LLC.*
- *Our malware analysts confirmed that three strains, BeaverTail, InvisibleFerret, and OtterCookie, are being used to spread malware via "interview malware lures" to unsuspecting cryptocurrency job applicants.*
- *The threat actor heavily utilizes AI-generated images to create profiles of "employees" for the three front crypto companies, employing "Remaker AI" (remaker[.]ai) for some of the AI-generated images.*
- *As part of the crypto attacks, the threat actors are heavily using GitHub, job listings, and freelancer websites.*

Okay. But that still doesn't convey what's going on. It takes some digging, but it turns out that North Korean hackers created and used two US front companies "Blocknovas LLC" and "Softglide LLC", registered in the states of New Mexico and New York, respectively. They faked being US companies, solicited US based employees into interviews, then infected those interviewees with malware that was then carried back to the prospective employee's current employers as a means to infect their organizations. And it worked. Yikes. So not only do employers now need to be very much on the lookout for spoofed fake employees, but anyone interviewing for a job change needs to be equally cautious and careful about the legitimacy of the company that says they might be interesting in hiring.

The new "Inception" AI jailbreak

Carnegie Mellon University's CERT Coordination Center posted the news of a new widespread vulnerability that affects pretty much all of the various GPT AI models. The title of their vulnerability report was *"Various GPT services are vulnerable to "Inception" jailbreak, allows for bypass of safety guardrails"* Here's what they explained:

Two systemic jailbreaks, affecting several generative AI services, have been discovered. These jailbreaks, when performed against AI services with the exact same syntax, result in a bypass of safety guardrails on affected systems and indicating a systemic weakness within many popular AI systems.

The first jailbreak, facilitated through prompting the AI to imagine a fictitious scenario, can then be adapted to a second scenario within the first one. Continued prompting to the AI within the second scenario's context can result in bypass of safety guardrails and allow the generation of malicious content. This jailbreak, named "Inception" by the reporter, affects:

- *ChatGPT (OpenAI)*
- *Claude (Anthropic)*
- *Copilot (Microsoft)*
- *DeepSeek*
- *Gemini (Google)*
- *Grok (Twitter/X)*
- *MetaAI (FaceBook)*
- *MistralAI*

The second jailbreak is facilitated through prompting the AI to answer a question with how it should not reply within a certain context. The AI can then be further prompted with requests to respond as normal, and the attacker can then pivot back and forth between illicit questions that bypass safety guardrails and normal prompts. This second jailbreak affects:

- *ChatGPT*
- *Claude*
- *Copilot*
- *DeepSeek*
- *Gemini*
- *Grok*
- *MistralAI*

These jailbreaks, while of low severity on their own, bypass the security and safety guidelines of all affected AI services, allowing an attacker to abuse them for instructions to create content on various illicit topics, such as controlled substances, weapons, phishing emails, and malware code generation. A motivated threat actor could exploit this jailbreak to achieve a variety of malicious actions. The systemic nature of these jailbreaks heightens the risk of such an attack. Additionally, the usage of legitimate services such as those affected by this jailbreak can function as a proxy, hiding a threat actor's malicious activity.

I don't even know how to respond to this other than to shake my head and understand just what a new wild west we've entered into.

One of the key coding lessons my past 50 years of programming computers has taught me is that if I'm not 100% completely certain how my code operates, it's unlikely to be correct, because there are so many more ways for it to be wrong than for it to be right.

Then I read about the bizarre ways it's possible to have conversations with these conversational AIs in ways that lead them to ignore the imperatives of their programming, and I also understand that no one is really completely certain how any of that works in the first place. And then I think of my own far simpler coding experiences and it becomes very clear that this incredibly fuzzy world of AI we're stepping into almost certainly has a far longer way to go before we're able to get a grip on it, than most people probably expect. I don't think we're even close.

SpinRite

One thing we all have in common is a concern for the integrity of our digitally stored data. In fact, it would not be an overstatement to say that I've made understanding and addressing the reliability of mass data storage my life's work, with the first half of my life invested in preparing for the second half where I've been able to do something about it and have created solutions to help recover data lost or seriously endangered for hundreds of thousands of PC users during the last 35 years.

Nearly two weeks ago the popular and respected Tom's Hardware website posted a piece under the heading *"Unpowered SSD endurance investigation finds severe data loss and performance issues"*. The start of that piece said: *"You may not know it, but SSDs will lose data after a period of time if they are simply left unplugged, which can be a serious threat to your data if you store backups or precious files on unplugged SSDs."* Not surprisingly, many of our listeners who are owners of SpinRite sent email wondering what I thought of the research Tom's Hardware shared. Before I share the rest of that piece, let's back up a bit.

Five years ago, early in the development of SpinRite 6.1, I created the ReadSpeed benchmark as a platform for verifying the operation of SpinRite's new low-level device drivers. The ReadSpeed benchmark takes an accurate measurement of a mass storage drive's performance at five locations: 0%, 25%, 50%, 75% and 100%. We all knew that spinning hard drives would perform much more slowly as we gradually moved toward their end, since track circumferences would be shortening, thus reducing their data transfer rate by half. But being entirely solid state, none of us expected to see any variation in SSD performance. But that's not what we found. Many of us discovered that the SSDs our PCs were using were much slower to read near their beginnings than anywhere else. What we discovered was that those regions which were only ever read and rarely or never written had become far slower to read over time. Since the front of these drives is where the operating system is written when it is first installed, we finally knew why, for years, PC users with solid state mass storage have been reporting that their systems seemed to have slowed down over time and to be running more slowly than when it was new. It turned out that it wasn't their imagination. Systems really do slow down because the reading performance of their solid state mass storage really is slowing down. And we also know this not just thanks to synthetic benchmarks like ReadSpeed or what's built into SpinRite, but because once SpinRite 6.1 allowed people to easily rewrite their SSDs, they reported that they could clearly feel the difference. Their machines were once again booting in seconds and the various annoying lags in its use had completely disappeared.

There have been a great many theories voiced to explain this. People get themselves all tangled up in the complexities of translation layers, wear leveling, block erasure, TRIMming and all of the many various technologies that have been layered on top of basic NAND storage cells in an effort to overcome those cells' inherent physical limitations. To my mind, donning my physicist's cap for a moment, there's really no mystery about why this is happening.

As I have described a couple of times in the past, flash NAND memory bits are just incredibly tiny electrostatic charge storage cells. They consist of a tiny bit of metal, which gives electrons a place to sit, surrounded by insulation, which keeps those electrons from wandering off. When we wish to change what's stored in that bit cell, we first create a high voltage. Remember that voltage is electrostatic pressure. So we create a high pressure that's able to break down the cell's insulation to inject some electrons across that insulation into that cell. The electrons that were injected under high pressure then remain there, trapped behind the cell's insulation. At that point, the magic of "field effect transistors" allows the effect of the resulting electrostatic field created by the charge stored in the cell to be sensed so we're able to later read out what was previously stored there.

Overall, this is an astonishingly effective technology; but it has one fundamental problem: We're deliberately abusing a cell's dielectric insulation whenever we use the brute force of high voltage to break it down and force electrons across the barrier it was designed to present. We want it to be a perfect insulator – except when we need it not to be. And over time, with repeated breakdown of its insulation, its insulating properties begin to falter and weaken with the barrier becoming slightly more porous to unintended electron migration.

With this bit of background, let's look at what Tom's Hardware wrote. Their piece said:

> *You may not know it, but SSDs will lose data after a period of time if they are simply left unplugged, which can be a serious threat to your data if you store backups or precious files on unplugged SSDs. A year-two update on the **"how long can SSDs store data unpowered"** video series is another reminder about the importance of regularly refreshing your backups with a bit of juice.*
>
> *The tests consist of storing data on an SSD and then leaving it unplugged for years to see the impact on the stored data. An SSD's endurance rating is calculated based on how long it can store data if left unplugged after a certain amount of data has been written, hence the importance of this testing.*
>
> *TechTuber HTWingNut is back with a report on his modest experiment involving a quartet of SATA SSDs. The key finding was that the two-year-old, well-worn drive exhibited noticeable performance degradation and was affected by a handful of corrupt files. These are signs that this particular SSD was on its way to silicon heaven. HTWingNut's video is an update on an episode from a year earlier, and further updates are promised.*
>
> *The four tested 'Leven JS-600' branded SSDs are basically bog-standard no-name units. HTWingNut says they are all TLC SSDs of 128GB capacity and rated to withstand 60 TB of written data. Every drive has 100GB of files containing random data, with hash values for all the content provided for later verification.*

I'll interrupt, again, to note that this is not how I would conduct such a test, since the file system's metadata that's being relied upon to access these files is sharing the same medium as the files it's managing. And you really don't want a filesystem involved at all. The right way to do this would be to use a pseudo-random function to generate a stream of data that would be written to the media. Then years later, use the same pseudo-random function to recreate the original data stream for bit-by-bit comparison with what is later read back.

But who am I to talk, since I didn't do any of that and this HTWingNut at least did what he did. So what we have from him is better than nothing. The article continues:

> *The two 'Fresh' sample drives have barely been used; perhaps only the 100GB data set was put there and verified, and that's it. Meanwhile, the two 'Worn' drives had been subjected to 280 terabytes of written data churn, much more than their rated 60 Terabytes Written (TBW) endurance rating..*
>
> *If you watch the previous year-one video, you will have seen there were no issues with either 'Worn' or 'Fresh' drives. However, time has now taken its toll. Let's take a look at the year-two samples in turn.*

For the 'Fresh' SSD tests, the data on this SSD, which hadn't been used or powered up for two years, was 100% good on initial inspection. All the data hashes verified, but it was noted that the verification time took a smidgen longer than two years previously. HD Sentinel tests also showed good, consistent performance for a SATA SSD.

Digging deeper, all isn't well, though. Firing up Crystal Disk Info, HTWingNut noted that this SSD had a Hardware ECC Recovered value of over 400. In other words, the disk's error correction had to step in to fix hundreds of data-based parity bits.

According to HTWingNut, seeing these errors means "the SSD is on its way out". Indeed, if there is anything iffy about your data storage integrity, it is at least a warning. However, the errors could also have something to do with the drive being left unpowered for two years.

As the worn SSD's data was being verified, there were already signs of performance degradation. The hashing audit eventually revealed that four files were corrupt (hash not matching). Looking at the elapsed time, it was observed that this operation astonishingly took over 4x longer, up from 10 minutes and 3 seconds to 42 minutes and 43 seconds.

Further investigations in HD Sentinel showed that three out of 10,000 sectors were bad and performance was 'spiky.' Returning to Crystal Disk Info, things look even worse. HTWingNut notes that the uncorrectable sectors count went from 0 to 12 on this drive, and the hardware ECC recovered value went from 11,745 before to 201,273 after tests on the day.

In summary, the year-one fresh and well-worn drives had no issues. However, the year-two heavily worn SSD had file corruption and performance was poor. The so-called fresh drive was still good, but ECC figures still raised concern. Come back in late 2025 for the next update from HTWingNut.

We also want to say this is a very small test sample, highlighted out of our interest in the topic rather than for its hard empirical data. I have also experienced SSD data loss after leaving a Mini PC unpowered for just six months or so at my pied-à-terre in Taiwan. On return, Windows refused to boot or be repaired, but a reformat and reinstall seemed to return everything to normal.

https://youtu.be/rx3Y5x6uzKQ I have a link to HTWingNut's YouTube video in the show notes.

Everything he found perfectly matches the model I've developed and shared about what's going on with our SSDs. The reason we see the performance drop when attempting to read data that was written long ago is that those electrostatic charges stored in the SSD's NAND bit cells have partially leaked away. This very slightly changes the voltages stored in the cells and forces the FLASH controller to work much harder to recover and reread the original data. We sense this by seeing the SSDs performance drop. If you ever notice a drop in SSD performance, that's the time to rewrite its data. You'll want to do so before that data becomes completely unreadable.

And the reason the problem is demonstrably worse on well-worn SSDs is that all of that prior writing further weakened the insulating dielectric which was keeping the electrons in place. So the leakage rate was significantly higher on those well-worn SSDs which were tending to lose their data faster.

One thing that hasn't been mentioned, which we also know from physics, is that temperature is crucially important. Several years ago we covered a piece of news that noted that offline SSDs stored in hot data centers tended to lose their data more quickly than the same SSDs stored in a cool environment. Heat inherently agitates electrons and increases the probability that one will make it across the cell's insulating barrier. So if you do have any offline SSDs or thumb drives where you have important data stored, I'd give them a full data rewrite pass with SpinRite at Level 3, then put them in a zip-lock bag in a refrigerator, or at least store them somewhere cool.

The reason why rewriting an SSD's existing data with SpinRite's Level 3 restores its factory-fresh performance, is that the act of rewriting an SSD literally restores the strength of its bits – which we now have additional and rather absolute proof, decay over time. ***Rewriting*** an SSD's data eliminates the uncertainty in the state of individual bits that can and does creep into our mass storage over time. Therefore, the speed with which an SSDs data can be read forms a highly visible and valuable proxy for the integrity with which the SSD's data is currently stored, readable and recoverable.

Let's see what other things our listeners have had on their minds this past week:

Closing the Loop: Feedback

John Canfield

Hi Steve, Like you, I had heard about this Windows Sandbox feature long ago and tried it briefly just to see it. Fast forward to about a year ago, I dug into it for a testing need I had and also was very impressed. I created a custom .wsb xml configuration with mapped folders to my PC, memory and CPU configs, and it sits on my PC to this day ready for use when the need arises. When you were describing the significant architectural capabilities and efficiencies that went into this feature, I can't help but think that this would be exactly what was needed for Windows 10X.

See Paul Thurrott's article, particularly the last sentence: ***"Worse, Microsoft hasn't addressed the single most important 10X feature, its planned ability to run Win32 apps in a container. Is that key work continuing?"***

https://www.thurrott.com/windows/windows-10/250230/microsoft-confirms-10x-death-will-bring-some-features-to-windows-10

Could Windows Sandbox have been developed for Win10X? or maybe the reverse, this feature existed before and someone said, hey let's use that for 10X 32 bit apps. Windows 11 came out in 2021, and Windows Sandbox was developed in 2018 according to your post. Those years

line up pretty well for one or the other to have happened.

All the usual praises, listening and watching back to the TechTV days, proud Spinrite owner, a joy to watch you and Leo every week. Best Regards, John

I chose John's question because it serves to highlight one of the reasons why Microsoft's implementation of Windows Sandbox is so economical.

The long-ago abandoned Windows 10X effort was Microsoft's ill-fated plan to wash away Windows long legacy of backward-compatibility. At one point they were planning to have a dual-screen Surface tablet PC and they wanted to move more toward a lean and mean OS, sort of like iPadOS. That meant essentially starting over from scratch with a new implementation of Windows, and among other things, that version of Windows would be dropping support for 32-bit Win32 apps.

Philosophically, I LOVE the idea of a complete reboot of Windows. One of the mixed blessings of today's Windows OS is that it still runs Win32 apps – and it probably always will because they cannot take that away. Too much legacy code depends upon it. Just look at how difficult it was for them to kill off Internet Explorer 6! IE6 refused to die because too many enterprise users had written code that would run nowhere else. And if you imagine that was true for IE6, just imagine trying to take away the Win32 API! Remember that Windows 7 included an XP Mode? XP Mode was a full VM that would allow Windows 7 users to run an instance of Windows XP. Why was Microsoft forced to include that? Specifically for backward compatibility, which serves as another example of the powerful drag created by Windows legacy code.

And in addition to the Win32 API, Windows also runs all of the other APIs that Microsoft keeps coming up with. I've lost track and count of the number of ways it's possible to author applications for Windows. And now they've added Linux subsystem support. One of Microsoft's biggest problems with Windows is they're unable to stop screwing around with it. They can't keep their hands off it. They're continually adding more stuff, but the critical need for backward compatibility means that they're never able to eliminate anything that came before.

They were finally able to drop support for 16-bit code when they moved to their 64-bit OS's, but even that was painful and they were only able to do so because Windows hadn't really gotten fully up to speed before everything had switched to 32 bits. So there wasn't all that much 16-bit code.

So, as I said, *"philosophically"*, I LOVE the idea of a massively simplified single API rewrite of Windows to create something truly lean and mean. But it's just a pipe dream. It's never going to happen, because what would remain would not be useful to anyone. And once smart people at Microsoft realized that, the Windows 10X project was dropped.

So John asked whether the Windows Sandbox might have in some way been a part of the Win10X project. But I cannot see how. What makes the Windows Sandbox so special is that it manages to surface an exact duplicate instance of the underlying OS in a sandboxed environment. It reuses the hosting OSes read-only files and even the underlying host OS's code loaded into read-only memory. That's its entire key. So if anything like the Sandbox were to run on top of Win10X, it could only be an exact clone of the OS it's running on, so it would be unable to, for example, support legacy APIs that had been removed through a host OS rewrite.

Antoine Choppin

Hello Steve, Thank you for Security Now! I had a question about Windows Sandbox you presented last week. You mentioned it uses a clever mechanism using links to static files to reduce the image size, which seems clever indeed, but made me wonder what would happen if the host OS had been compromised and some files (supposedly read-only) had been modified somehow. In that case, I guess the sandbox would be compromised the same way, which means it is not as isolated as one could think. Curious to hear your thoughts on this. Thanks again for the great podcast! Antoine

I'd say that Antoine is completely correct. And it would likely go even further. Since we know that the Windows Sandbox also conserves its usage of RAM by mapping the underlying host OS memory footprint into its own memory space, any malware that operated by "hooking" kernel API functions in RAM – which we know is something malware commonly does – would inherently duplicate those hooks, and the same OS compromise would appear inside the sandboxed OS.

So Antoine's point is a good one. And it's an important distinction between a sandbox and a full virtual machine. As Leo noted last week, the Sandbox solution is closely aligned with the concept of containers which share many of the same properties. Neither the Sandbox nor Containers contain an entire isolated instance of an operating system. They use Hyper-V virtualization to create and enforce "containment" of the code they host, but they're running on top of their containing host. So neither Windows Containers nor the Windows Sandbox are isolated from underlying host problems. Only a full stand-alone virtual machine would provide that. But that level of isolation code at the cost of significant host platform resource consumption with a full virtual drive and much more RAM consumption. All of these various technologies are interesting and powerful and each has its place.

Brian

Hi Steve, Love the show and a proud owner of Spin-Rite. I know this may be a bleak question, but would you consider open-sourcing Spin-Rite upon your eventual but hopefully distant passing? It's an excellent product and I just don't have faith that people will put this kind of effort into something like this again. I'd love to see Spin-Rite live on and continue to keep up with hard drive technology into the far future. Thanks! Brian (you can use my first name if you ever mention this on air)

I don't consider this to be a bleak question at all. I consider it to be practical and flattering. Our listeners here would have no way of knowing that I have formally stated several times in GRC's newsgroup forums that it is my intention to release all of my work – the source code for everything I've ever written – into the public domain once my own commercial interests are no longer connected to it.

Ideally, this would occur at some point when I still have some cognitive faculties available so that I could shepherd that code into the world and be available to answer any questions that would doubtless arise. So I very much look forward to that day since I think that would be a lot of fun. The bottom line is that, yes, once I hang up my spurs or am struck by lightning, everything I've created will be released to the public and I would be honored if there was interest in keeping it alive and growing into the future in whatever form made sense.

Gaelin

Hello Steve, In episode 1019, you were talking about the constant internet spam and brute forcing going on; it is so much worse than you stated. I have ssh open on my homelab so that I can manage it remotely, with Fail2Ban configured. Fail2Ban monitors auth logs and can do automated actions based on successive failures. I have Fail2Ban setup to ban the IP of anyone who has 2 failed login attempts for 3 hours, then ban anyone with 2 bans in the same day for a year. As this lab is only used by a close friend and myself, and we both use keys to authenticate, it is unlikely for us to ever have a failed login attempt. I set it up with a discord bot to automatically notify me of bans and send me daily reports on ban counts, and it is crazy to watch. I have seen days with up to 5000 UNIQUE IPs banned, normally it is around 300-500. I see a failed login attempt around every 2-3 minutes 24/7/365, not all of them end up banned because some of the bots space their logins out a lot. I have banned around 26,000 unique IPs and at any moment have around 4,000 banned. I highly recommend that anyone hosting publicly accessible SSH install Fail2Ban, even with just default settings ssh.
Thanks for the podcast! Gaelin

This was a great data point, and it not only supports what we were talking about four weeks ago during podcast 1019, but also more recently, when I was talking about the fact that typical network monitoring is only looking at what gets inside the network. While that's certainly of the most concern, there's still the fact that we don't know what we don't know.

The fact that Gaelin has witnessed this first hand has doubtless altered his behavior in a healthy direction. It will serve to inform him about just what a jungle it is out there and the degree to which he really can never afford to take his own security for granted. Say, for example, that he was still relying upon a username and password for protection. If he didn't already know better, seeing the truth about how much attention his own SSH server is drawing would doubtless motivate him to take the time to be as secure as he could possibly be.

Like Gaelin, I've looked at my own external bandwidth logs and what's going on, as he said 24/7/365 is truly harrowing. I mean it's insane. We talked a few podcasts ago about the abuse of login attempts to Microsoft Outlook, and how wrong it feels that they are not providing better abuse protection. Everyone knows that credential stuffing attacks have grown to become one of the major threats, yet Microsoft only offers geofencing for their enterprise users.

A few podcasts ago I took the opportunity to rave about my absolute favorite SSH client and server, BitVise. Many of our listeners wrote to let me know that Windows already has SSH client and server solutions built-in. And that's absolutely true. Windows now offers the industry standard setting OpenSSH server. So thanks to our listeners for noting that for me.

But Windows doesn't have BitVise built-in. In addition to having an extremely pleasant zero-learning-curve graphical user interface, I have my BitVise server instances configured to only consider accepting incoming connections from IPs located in the United States. And within the US, since connecting to the BitVise SSH server with the BitVise client is 100% reliable, a single failure to authenticate from within the US permanently blacklists the IP. And must so that I'm not locked out in the event that I fumble-finger the connection at the client end, I have permanent whitelist IP overrides for the two IPs I would probably be connecting from. As I've mentioned previously, my two cable modem IPs are extremely static. And all of that is after configuring the server to only accept authentication via a public/private key exchange challenge. Finally, all of that was done with a few clicks of a mouse while browsing the BitVise user interface. So, much as I strongly prefer "living off the land" solutions using what's already present. In this case I'm not giving up BitVise for anything. It retains my highest possible recommendation.

And speaking of the utter mess that the Internet is outside of our walls...

Matt Davis

Hi Steve,

I wanted to share a bit of an unexpected side effect that I experienced a few months ago when Lets Encrypt stepped up from single-perspective issuance and started requiring a second perspective.

I run a small web hosting business on the side for a few clients, and one client called me one morning to report that her website was showing the big scary red Certificate Warning page in Chrome. I took a look and sure enough, her Lets Encrypt certificate had expired the evening before. As you know, all LE certificates should be renewing automatically through the ACME protocol.

After troubleshooting this problem for over an hour, I eventually realized what was going on. This client runs a small local photography business in the USA. In working to secure her Wordpress site, we made a quick and easy decision - she didn't need any web traffic from China, Russia, or any other country banging at her digital door. If the person trying to access the site wasn't in her local area, or even in the USA, they simply had no business being there.

*So we set up Cloudflare to block all traffic from all 194 other countries - it was of no use to her, and it eliminated **massive** amounts of bot traffic, image theft hotlinking, AI scraping, Wordpress login attempts, and other shenanigans. After implementing that rule, requests to her site (again, a local photography business!) dropped over 95%, and bandwidth use reduced by even more than that.*

*However, now with ACME challenges coming from random countries around the globe, I've had to take steps to whitelist those Lets-Encrypt challenges no matter where they come from. Multi-perspective issuance has **reduced** this site's security, as our WAF is now forced to allow certain traffic from any country at any time. This may be an unusual example, but when a website really doesn't need to be global, you can easily reduce your attack surface through geoIP firewall rules and other limitations... or at least you used to be able to. Thanks, Matt.*

This is such a wonderful real-life example of the mixed-blessing consequences of increasing security. Whenever we tighten anything down to prevent its abuse we run the risk of triggering false-positive blocks. In my own example of super-tightly locking down my own access to my BitVise SSH server instances, I was acutely aware that, yes, there would be some risk that I might lock myself out of my own server. But that was a balance that I judged to easily be worth the risk.

In the instance of Multi-Perspective Issuance Corroboration, we've only heard from one of our listeners – and thanks for sharing that, Matt, what a great story. But it's not difficult to imagine that many thousands of other ACME-based certificates were also probably recently similarly impacted. And Matt's right that by needing to allow a subset of queries from anywhere through to his client's server, he's been forced to reduce its overall security.

And if Matt were to tighten down on the class of foreign queries that were allowed to reach the server, so that only those qualifying were allowed, then any change that Let's Encrypt might make to their query protocol could again cause a breakage. We're in a world of tradeoffs.

One thought I had, and I imagine this probably occurred to Matt, was that Let's Encrypt queries over port 80 using HTTP rather than HTTPS. The good news is that pretty much nothing else uses port 80 anymore. We were recently talking about Cloudflare dropping all API support over port 80. I haven't looked at Cloudflare's country-based filtering. But if it were possible to block all port 443 access from everywhere other than the US, that ought to restore much of the benefit of a full blanket block.

So that would mean that only traffic incoming to port 80 would be allowed from anywhere. Then, since Let's Encrypt's ACME protocol always and only looks for its domain control authentication token in the "acme-challenge" subdirectory of the ".well-known" root directory, it would probably be possible to set up an .htaccess or web-config rule to only allow queries over port 80 to that one directory. That ought to allow Let's Encrypt to obtain what it needs over port 80, incoming from anywhere in the world, while not giving any of the rest of the non-US world anything it might find interesting.

Daryl in Kansas

Steve, I'm a SpinRite site license guy. [Much appreciated, Daryl] *I listen to every Security Now episode! How safe is the "trust this computer" option for websites when you're at home on your own network? (I use a Chromebox for extra security) Do you click yes, or "let sleeping dogs lie"? Thanks for Security Now, and Hi to Leo! :)*

That's a terrific question because what's going on beneath the surface is not at all obvious from the question itself. As we know, each of our web browser's queries to remote websites stand alone. That means that unless something explicit is done, there's no way for a remote website to know who any given query is coming from. That something explicit that is now always done is that any web browser query that's made, which does not include a browser cookie, is replied to with a new unique cookie, so that all subsequent queries which issue from that web browser will automatically be "tagged" with that new unique cookie, since that browser cookie will always be returned. So the first thing to appreciate is that all of the web browsers that are querying remote web servers – if they don't already have one – are each given a unique cookie so that the remote site has some means of telling them apart.

The next important point is that if a specific user identifies themselves to that remote website by logging onto it using some credentials, it's the ongoing presence of this cookie that serves to keep them logged in.

Next, it's probably always possible to deliberately and explicitly logout of any website. There's almost always going to be some "logout" option, generally, by growing convention, in the upper right hand corner of the website's pages. But the question is, what happens if you do not remember to logout? Many websites don't care at all how long you've been gone. When you return you'll still be logged into that site. And the only reason you'll still be logged into that site is that your web browser has remembered and still has the cookie it received then last time you logged in. GRC used XenForo for its various web forums and I cannot recall the last time I was asked to login there. That's a convenience, since I'm the only one using any of the computers where I'm logged into our forums. So I'm able to just go to forums.grc.com and pick up right where I left off without any need to prove to the forum server who I am.

But what if multiple people use the same computer? Or what if you're logging in at an Internet Café or a public library? In that case you would not want your logon to be so persistent. And THAT is what the "Trust This Computer" checkbox, which often accompanies a logon page, is all about.

Cookies all come with an **optional** expiration date. If that date is ever reached, the web browser will no longer honor that cookie. Instead it will simply delete it. But I mentioned that the expiration date is optional. If a cookie is given to a web browser without any expiration date then that cookie is deliberately never written – in any way – to any form of persistent physical storage. It is only ever retained in RAM. That means that once the web browser application is closed, the values of any of the non-expiration-dated cookies it may have received while it was running will be lost forever. And that is the beauty of NOT having the "Trust This Computer" checkbox checked when you log into a website.

When logging in with that checkbox unchecked, any logon authentication cookie your browser receives will have no expiration date set. So it will be "ephemeral" and your logged in identity will be deliberately lost when you close the web browser application.

So Darly in Kansas, you asked *"How safe is the "trust this computer" option for websites when you're at home on your own network?"* Only you can really answer that. But now you probably can, since you should have a good understanding of exactly what that means. It boils down to whether anyone else might have physical access to any computer where your prior logons would be persistent because you had enabled the "Trust This Computer" option which will have created persistent logon sessions.

If you are the only person who has access to any computers where you might have left a site logged on, then remaining logged on is likely a convenience that would have no downsides. But if others might use a computer where you were left logged onto a site which you would prefer they not gain access to under your account, and since you might easily forget to explicitly logout after using that site, then logging in, in the first place, with "Trust This Computer" disabled, would mean that you'll be automatically logged out when the browser is closed or the computer is turned off.

Angus MacKinnon

After reading the following, what would you recommend? I am a backblaze customer.

Yikes! Angus's note included a link to a document from the website of Morpheus Research: https://www.morpheus-research.com/backblaze/ The Morpheus Research page which was published late Thursday is titled: *"Backblaze: A Loss-Making Data Storage Business Mired in Lawsuits, Sham Accounting, and Brazen Insider Dumping"* It leads off with a bullet-pointed summary of past management conduct that's enough to make your hair curl and then fall out. It's really something.

The name Backblaze is quite familiar. They've been around for 18 years, having been founded in 2007. I wasn't aware that the company had gone public four years ago in 2021, but at that time they used their IPO to raise $100 million dollars. According to the research article, it sounds like the founders did not place much faith in their own company's future since as soon as the IPO lock-up expired in April of 2022, they began aggressively cashing out their publicly traded stock by selling 10,000 shares per day. And since the IPO, the share price has dropped by 71%. I'm mentioning all this because, if we are to take this report at face value, it doesn't appear that Backblaze is currently very healthy. They may have customers and assets and cash flow that some bigger fish might want. So they might be purchased. But this report has clearly unnerved our listener, Angus, who wonders what I would recommend. The same report mentioned that Backblaze was losing, and had lost, many customers to Wasabi. Since they were a past sponsor of the TWiT network their name was familiar to me – so I'd recommend checking them out.

Preventing Windows Sandbox Abuse

Last week's "Windows Sandbox" podcast reminded us that everyone with Windows 10 or 11 – with the exception of Home edition users – has access to a very nifty Windows execution environment specifically designed to allow users to safely experiment with "throwaway" programs, installations, files and anything else, without having any impact on their primary Windows OS installation. And, moreover, I was very impressed with Microsoft's surprisingly efficient and economical implementation which got so many things right.

One interesting feature of Windows Sandbox which I believe I mentioned in passing last week, is that Windows Defender is disabled by default within the Sandbox and it cannot be enabled via either the GUI or PowerShell commands. This decision was presumably made because running Defender inside the Sandbox would slow everything down, because users might specifically wish to run things that would cause Defender to freak out, quarantine and delete their files, and because the entire point of the Sandbox is that it's a place where terror may safely reign with full confinement.

Unfortunately, it would probably come as no surprise to anyone who's been following this podcast for long to learn that bad guys have figured out how to take up residence in Windows Sandbox as a means of obtaining secret persistence within Windows systems while being hidden from the instance of Windows Defender or any other A/V scanning, that may be patrolling the grounds outside of the sandbox. So let's take a closer look at how Windows Sandbox is being abused and what that means. And then we're going to examine what can be done to prevent its abuse whether a user wishes to use Windows Sandbox or not.

I'm going to start by sharing a nice overview of the problem which appeared in the Risky Business newsletter. That newsletter was headlined "Chinese APT abuses Windows Sandbox to go invisible on infected hosts", and Catalin wrote:

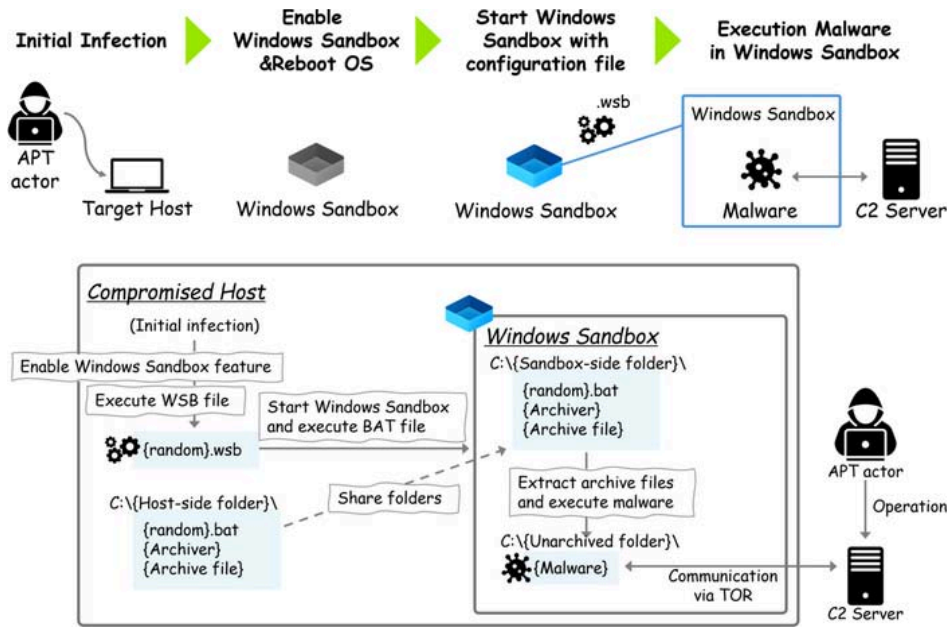
A Chinese cyber-espionage group named MirrorFace (aka Earth Kasha, APT10) is abusing the Windows Sandbox virtual environment to hide the execution of its malware on infected systems. Attacks incorporating Windows Sandbox have been taking place since 2023 and represent the first known case of Windows Sandbox abuse since its release in December 2018.

As the name hints, the feature allows Windows users to start an isolated sandbox where they can temporarily install/test apps and then shut down the virtual environment without impacting the main OS and their data. It functions as a virtual machine, but it doesn't have all the bulky features of a VM, it's light, super fast, easy to start and use.

Abuse of this feature sounds implausible because Windows Sandbox support is disabled by default, and when a Sandbox is started, it runs in a window in the user's foreground. But according to reports from the Japanese government and ESET, MirrorFace has found a way around these limitations. The group gains an initial footholds on compromised networks, enables Windows Sandbox, restarts systems, then silently launches Windows Sandbox instances that don't appear on the screen. This is accomplished by launching the Sandbox via Task Scheduler under a different account from the user's current one, so the Sandbox UI never appears to the logged-on user.

The MirrorFace operators drop malware in a folder on the infected systems then use Windows Sandbox .WSB configuration files to share access to that folder to the Sandbox, grant the sandbox network access, then configure one of the malicious files to automatically run when

the Sandbox is executed.



Since Windows Sandbox environments cannot run Defender, nothing that happens inside is either logged or detected. This allows the attacker to install malware and open a hidden backdoor inside that system and a victim company's network. Japanese security firm ITOCHU explains how blind companies can become against Windows Sandbox-based attacks, writing:

"Since the malware in Windows Sandbox operates according to the WSB file's configuration, it can access files on the host machine. However, because the files are accessed from the sandbox, activity is not logged by monitoring tools running on the host system."

The technique used by MirrorFace seems to be an evolved version of a technique first documented by security researcher Lloyd Davies back in 2020. ITOCHU researchers say the abuse can go a few steps further since new features are constantly being added to Windows Sandbox. For example, the Windows Sandbox can now share clipboard, audio, and video input with the base OS. The Windows Sandbox can now also be started via command line arguments using the new "wsb.exe" command, which removes the need for WSB configuration files—artifacts that security firms could use to detect possible abuse.

The technique is incredibly simple to automate, even for low to mid-tier skilled malware developers. Once detailed in these reports, it is likely to spread to other groups.

The first to jump on and abuse this technique are likely ransomware gangs. Some groups are already using something similar. At least half a dozen ransomware groups have been spotted installing bulky VM software on infected hosts just to start the VM and send victim files to be encrypted inside, where security tools don't have access to spot the ongoing encryption.

Since Windows Sandbox is built-in and present on all Windows 10 and Windows 11 systems, and the app's file is signed by Microsoft itself, abusing it is likely easier and safer. ITOCHU has published some monitoring and infection remediation advice to detect this technique, but the cat is out of the bag, and further and broader abuse is now expected to start taking place.

One thing that's very interesting is the observation that the Windows Sandbox is able to launch and run under a different user account so that the foreground user never sees any indication that this is happening in the background. And here, the inherent efficiency of Windows Sandbox which so impressed me last week actually works against us, since its lightweight nature means a user would be less likely to wonder where all their free RAM went because it wouldn't be consuming much.

Also, the default-enabled clipboard sharing is a bit chilling, since it would be a bit like having a malicious instance of Windows Recall running unseen in the background, capturing anything the foreground user might temporarily place onto their clipboard... such as a cryptocurrency wallet address.

I was curious to see what this researcher "Lloyd Davies" came up with five years ago in 2020. Whatever it was, Microsoft apparently blew it off without a second thought since we're now five years downstream of that and Windows Sandbox is still here and completely abuse prone.

Five years ago, under his headline *"Weaponizing Windows Sandbox To Bypass Defender"* Lloyd Davies wrote:

> *This short blog post may be useful for a Red Team living-off-the-land for the execution of payloads on a machine where Windows Sandbox can be enabled; Windows Sandbox is designed to work this way - no exploitation of anything is covered in this post. With this technique in terms of executing within a VM, we don't need to load an external ISO onto the machine, as all of this is handled by the sandbox. In my research, the Sandbox .wsb configuration file was not inspected or blacklisted on any major EDR or AV.*
>
> *At the tail end of last year, Microsoft introduced a new feature named Windows Sandbox (WSB for short). Windows Sandbox allows you to quickly, within 15 seconds, create a disposable Hyper-V based Virtual Machine with all of the qualities a familiar VM would have such as clipboard sharing, mapping directories etc. The sandbox is also the underlay for Microsoft Defender Application Guard (WDAG), for dynamic analysis on Hyper-V enabled hosts and can be enabled on any Windows 10 Pro, Enterprise or Education machine - making this perfect as a living off the land technique.*
>
> *The TL;DR of this technique is to craft a .wsb that can be executed on an endpoint, which mounts the user's file-system, allowing us to execute the implant inside a hidden VM and bypass any AV/EDR that's on the host. The WSB configuration also seems to be bypassing Windows Defender on the host where it's executed. It's not incredibly complicated but could prove useful in an engagement.*

Lloyd then proceeds to talk about and document the various ways very powerful .WSB files can be created to give a malicious sandbox all the power it might need on the user's system, all while always remaining completely hidden and undetectable. He concludes his observations by writing:

> *A similar technique has been used by the infamous Maze and Ragnar locker threat actors in recent times; however, they've installed 3rd party virtualisation suites such as VMWare & VBox. Using Windows Sandbox bypasses the requirement for this software to be installed. To complement this technique, I created a simple Go program to find drives automatically and mount network shares that include them as mapped folders, and to then generate a WSB based on this.*

https://blog-en.itochuci.co.jp/entry/2025/03/12/140000

I have a link in the show notes to an English language translation of the talk that was given last January in Japanese by the ITOCHU researchers. Among the many things they have noted is that with the introduction of Windows 11 Microsoft has enhancing the Sandbox's features in ways that allow for additional abuses. They wrote:

> *The changes to Windows Sandbox after the Windows 11 update are as follows: Addition of the wsb.exe command, enabling sandbox execution via the command line, background execution of the sandbox, and the ability to modify certain settings via the GUI. These recent feature updates may make it more difficult to detect attacks leveraging Windows Sandbox. The key reasons for this are as follows:*
>
> - *Background execution of Windows Sandbox — previously, in Windows 10 and early versions of Windows 11, Windows Sandbox always ran as a foreground GUI application. However, with the new **wsb.exe start** command, it can now run in the background. As a result, the sandbox can be launched without user awareness, and its window remains hidden until the **wsb.exe connect** command is executed.*
>
> - *Sandbox execution without a WSB file — The updated **wsb.exe** command allows sandbox configurations to be set via command-line arguments. Previously, WSB files were an important forensic artifact during investigations, but this change increases the risk of leaving no trace of sandbox usage.*
>
> - *Persistent data inside the sandbox — In earlier versions, closing the Windows Sandbox window would terminate the process and delete all data within the environment. However, after the update, closing the window does not stop the sandbox, and its data remains intact. To delete data, the sandbox must be explicitly stopped using the **wsb.exe stop** command or terminated by shutting down the host machine. This change significantly increases the potential for long-term attacker operations within the sandbox.*
>
> *Given these updates, security researchers must carefully verify whether such feature changes improve convenience for attackers and implement appropriate countermeasures when new functionalities are introduced.*

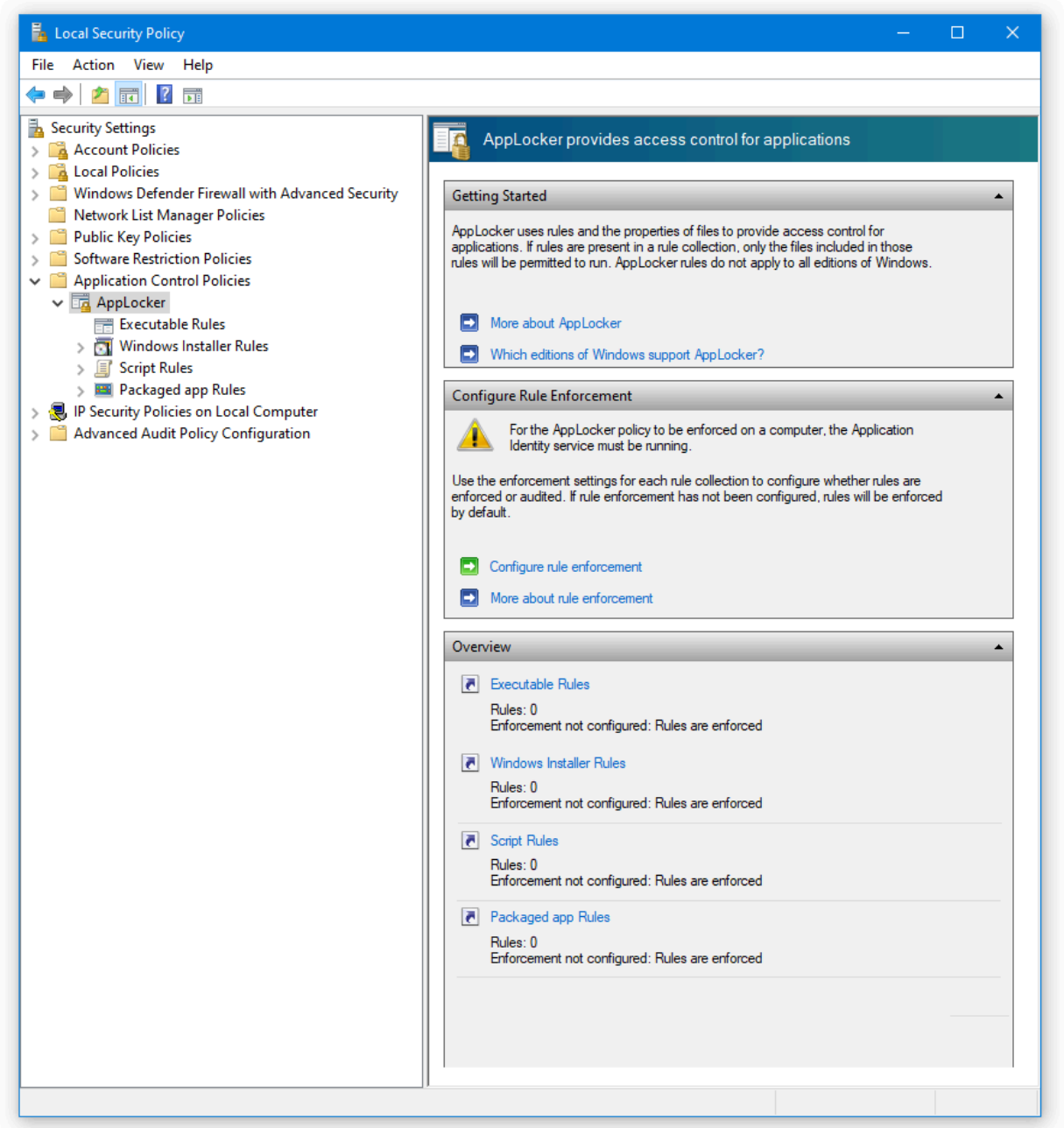
I titled today's podcast *"Preventing Windows Sandbox Abuse"* because, having now explored the dark side of this otherwise truly useful and nifty Windows Sandbox feature, if it's not something that its user will be actively using, it might be worth considering taking some measures to neuter it so that it cannot be abused behind its users back.

My number one favorite way to do this would be to disable a system's virtual machine extensions capabilities at the pre-boot firmware level. I recently learned that the BIOS settings backup battery on the aging Gigabyte motherboard of my older Win7 machine had died. My neighborhood had a planned day-long power outage while our local power company's equipment was replaced. When I fired my machine up after having it shutdown for the day, I quickly saw that it had lost its time of day and date clock. So I rebooted, went into the BIOS and set that correctly. Some time later, when I attempted to launch a VirtualBox virtual machine, I received an error that VirtualBox was unable to operate without the Intel Virtualization Technology – abbreviated as VT-X – enabled in the system's BIOS. I mentioned last week that the same is true for Windows Sandbox. The Microsoft Hyper-V virtualization technology the Sandbox depends upon is, in turn, dependent upon having Intel's Virtualization Technology enabled.

So the absolute best protection for anyone who does **not** routinely use either the Windows Sandbox nor any of the many other various virtualization systems, since all of those are now known to be prone to abuse – and especially Windows Sandbox – would be to simply run without the Intel VT-X extension enabled. No VT-X means no virtualization funny business. Doing this will have zero impact upon Windows operation and it will completely shut down any chance of abuse.

If you do need to run virtual machines other than Windows Sandbox you'll need to have the Intel VT-X extensions enabled in your machine's firmware. Enabling Windows Sandbox requires admin privileges. But we know that doesn't present much of a barrier to malware, since pretty much everything bad that malware does requires admin privileges. And we know that elevation of privilege exploits are constantly being uncovered.

The solution for someone who wishes to prevent any "behind their back" exploitation of Windows Sandbox and for whom disabling all use of VM technology via the VT-X extension is not an option, Windows AppLocker is probably the next best solution.



AppLocker can either be configured in a managed enterprise setting through group policies or on a local machine using the Local Security Policy snap-in. The use of AppLocker is straightforward and many How-To's exist on the Internet for anyone who wants to take that approach.

Under Windows 10 or 11 you'll want to block the execution of the WindowsSandbox.exe executable program in the System32 directory: %SYSTEM32%\WindowsSandbox.exe. And additionally under Windows 11 you would want to prevent the wsb.exe command from being used.

Once any use of those have been foreclosed, anything that tries to crawl into your machine and set up shop behind your back using the Windows Sandbox will be out of luck.