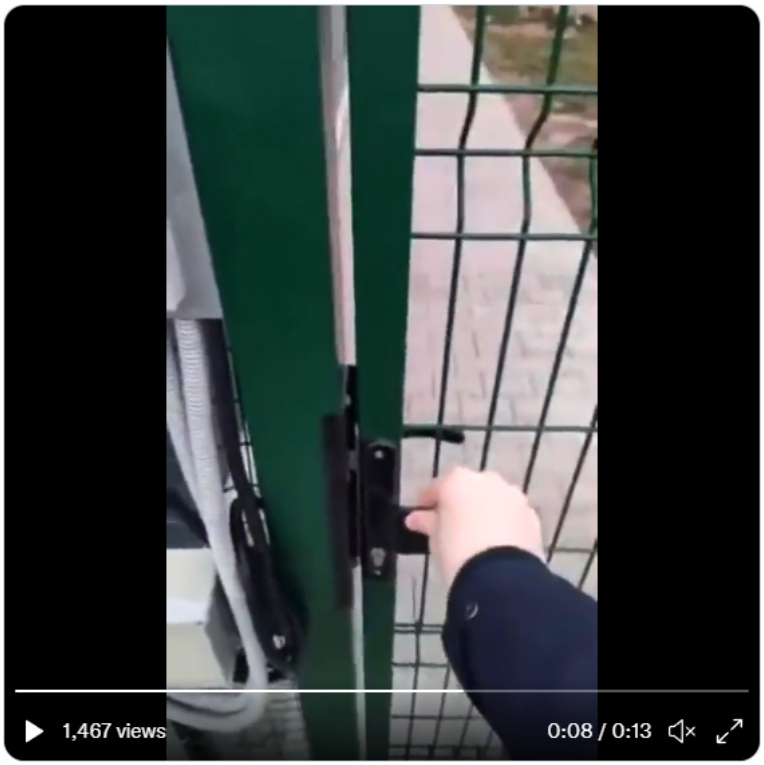
Security Now! #893 - 10-18-22

Password Change Automation

This week on Security Now!

This week we examine several more serious Microsoft security failures which have just come to light, and a new useful Windows security feature that was just added. The new Passkeys logon technology received its own website to monitor its progress, and Cloudflare logs another record breaking DDoS attack. Signal drops its legacy support for SMS/MMS on Android, Fortinet attempts to keep a new bad authentication bypass quiet, the White House proposes work on an IoT cybersecurity seal of approval, and the US Treasury department levies a hefty fine against a cryptocurrency exchange for not caring who they send money to. I have some updates on SpinRite, my just-discovered ZimaBoard and two pieces of listener feedback. Then we're going to finish by examining a new standardized means of accessing websites' password change pages. And we also have our first-ever **Security Now VIDEO of the Week...**

The Security Gate



https://grc.sc/gate
https://twitter.com/HamzahBatha/status/1579794162807103490

Security News

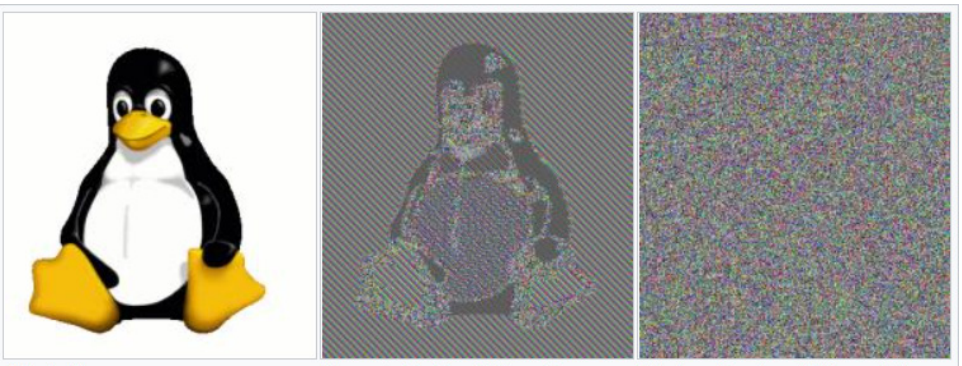
"Won't Fix"

I gave this first discussion the title of "Won't Fix" because that's what Microsoft told the guys who noted and reported their discovery. They said "Thanks very much, but we're going to leave it as is." So last Friday, the pattern of Microsoft deliberately choosing not to fix a latent, well understood, and potentially serious security vulnerability repeated itself.

Many years ago, when this podcast was first laying out the fundamentals of encryption, we talked about cipher modes. Any practical encryption system starts with an underlying symmetric cipher. The one the industry has settled upon today is the Rijndael Cipher which was chosen to be the AES standard. It's a 128-bit block cipher, meaning that it takes a block of 128 bits, which is 16 bytes, at a time and under the influence of a key which is typically 256 bits, it arranges to map every possible input combination of those 128 bits into a different output combination of 128 bits.

As long as the key remains the same, every time the same 128 bits is presented, the same different 128 bits is produced. This is required for a cipher to be useful. Obviously it must be deterministic and not generate random outputs. But this determinism also poses a problem which we've talked about several times in the past. If the same plaintext input block always produces the same ciphertext output block, then someone examining the encrypted ciphertext output, who sees identical output blocks appearing, instantly knows that the input blocks were also identical. They may not know what they were, but given sufficient time, statistical analysis can leak significant information. And if **any** of the input text is known, like standard query headers, packet protocol data, boilerplate or any other overhead, then someone examining output blocks can see what those known input blocks encrypt to.

This simple and straightforward yet ineffective mode and method of encryption is known as Electronic Codebook or ECB and no one uses it for encryption specifically because it is clearly and obviously insecure. (By now you can probably guess where this is going.) The most famous clear example of the failure of Electronic Codebook mode to effectively protect the secrecy of data is the classic demonstration of the image of the Linux Penguin:



Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

The third image is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the third image does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as "random-looking".[*citation needed*]

From Wikipedia: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

The center image is the result of using the simplistic electronic codebook (ECB) encryption on the image of the Linux Penguin. As we see, the "encryption" (in quotes) leaves a clear impression of the original image.

What we actually want is shown in the right-most of the three panes where the result is pure noise without any vestigial remnant of the original image. **Any cipher mode other than the simplistic ECB, results in something like the third image:** Actual encryption which does not leak information about the unencrypted plaintext. I won't delve into great detail about the other encryption modes since we have carefully and fully covered this before, and the Wikipedia link about this will refresh anyone's memory if they're curious. But the crucial weakness of simple electronic codebook encipherment is that each encrypted block stands alone. The good news is, this is easily resolved. Every one of the other proper encryption modes solves this simply by chaining. The most famous of these modes is CBC, which stands for Cipher Block Chaining. CBC simply XORs the result of the previous encrypted block with the plaintext to be encrypted by the next block. That's all it takes. By chaining the encrypted result into the next encryption, blocks no longer stand alone. Each block is affected by all previous blocks.

So what happened with Microsoft last Friday? The company now known as "WithSecure" which was formally F-Secure Business, published their distressing summary of events under the title: "Flaw in Microsoft Office 365 Message Encryption could expose email contents to attackers" They explained:

> *Adversaries can exploit the flaw, for which there is no patch available, to obtain information that could lead to a full or partial information disclosure.*
>
> *Helsinki, Finland – October 14, 2022: Today, WithSecure™ (formerly known as F-Secure Business) published a security advisory warning organizations of a security flaw in Microsoft Office 365 Message Encryption (OME).*
>
> *OME, which is used by organizations to send encrypted emails internally and externally, utilizes the Electronic Codebook (ECB) implementation – a mode of operation known to leak certain structural information about messages.*
>
> *Attackers able to obtain enough OME emails could use the leaked information to partially or fully infer the contents of the messages by analyzing the location and frequency of repeated patterns in individual messages, and then matching these patterns to ones found in other OME emails and files.*
>
> *Harry Sintonen, WithSecure's security researcher, who discovered the issue, said: "Attackers who are able to get their hands on multiple messages can use the leaked ECB info to figure out the encrypted contents. More emails make this process easier and more accurate, so it's something attackers can perform after getting their hands on e-mail archives stolen during a data breach, or by breaking into someone's email account, e-mail server or gaining access to backups."*

According to the advisory, the analysis can be done offline, meaning an attacker could compromise backlogs or archives of previous messages. Unfortunately, organizations have no way to prevent an attacker that comes into possession of affected emails from compromising its contents using the method outlined in Sintonen's advisory.

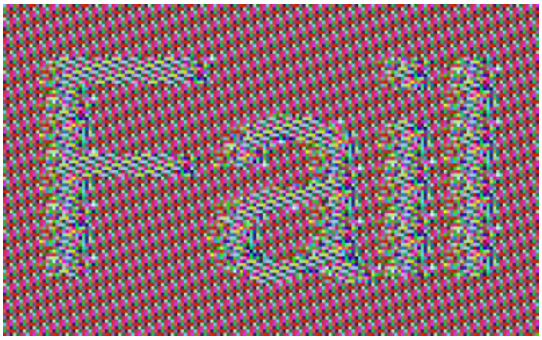
The advisory also highlights that no knowledge of the encryption keys is needed to conduct the analysis, and that use of a Bring Your Own Key (BYOK) scheme does not remedy the problem.

Sintonen shared his research with Microsoft in January 2022. While Microsoft acknowledged the problem and paid Sintonen via their vulnerability reward program, they opted not to issue a fix. While organizations can mitigate the problem simply by not using the feature, it does not address the risks of adversaries gaining access to existing emails encrypted with OME.

Sintonen said: "Any organization with personnel that used OME to encrypt emails are basically stuck with this problem. For some, such as those that have confidentiality requirements put into contracts or local regulations, this could create some issues. And then of course, there's questions about the impact this data could have in the event it's actually stolen, which makes it a significant concern for organizations."

Because there is no fix from Microsoft or a more secure mode of operation available to email admins or users, WithSecure recommends avoiding the use of OME as a means of ensuring the confidentiality of emails.

And the trouble is not just theoretical. WithSecure's technical write-up placed an image in an Office 365 message, encrypted and sent it using their OME – Office Message Encryption – and this was the result:



As we've seen over and over, Microsoft's industry dominance is so complete that the details of what they do no longer matter. This won't cause them to lose a single customer, and they know it. So, why bother fixing it? Just call it "encrypted" and figure that it's "encrypted enough."

This begs the question of how this could have ever happened in the first place? How do you put someone in charge of adding encryption to Office 365's eMail system who, by any objective measure, must have been utterly incompetent to do so? Again, the evidence suggests that even that just doesn't matter any longer. Microsoft's dominance means that it's too big to care.

Unfortunately, we're not yet quite through with Microsoft for the week...

Malicious Kernel Drivers

ArsTechnica's coverage of this had the headline: *"How a Microsoft blunder opened millions of PCs to potent malware attacks"* with the subhead: *"Microsoft said Windows automatically blocked dangerous drivers. It didn't."*

Before we get into the details of what was discovered, recall that this is something we've covered in the past. The issue is that kernel drivers, by their nature and position in the system, run with the highest available privileges — in the Windows kernel, where they can do anything and everything. So they must be absolutely trusted. But they also contain complex code which requires a level of attention to detail that can be lacking. So, otherwise perfectly benign drivers can have identified exploits. Such drivers will originate from valid reputable sources and bear their reputable publisher's digital signatures. So they're entirely valid. And signatures never expire. If a signature is valid when it was signed it remains valid. But what if a problem is identified in an otherwise valid driver; a problem that's maliciously exploitable to gain privilege elevation? Once this becomes public knowledge, bad guys can bring and install one of these valid drivers into a system that otherwise has no need for them. It's like their own private backdoor. This exploit technique is known as a BYOVD attack — Bring Your Own Vulnerable Driver — and it's a real concern since kernel drivers are no less subject to bugs than anything else.

The only solution is to prevent known vulnerable drivers from being accepted and loaded into Windows. And that requires that all such known vulnerable drivers be black listed. Of course, this critical protection strategy is only as effective as the list of known vulnerable drivers is kept current. And so begins our story.

As ArsTechnica's Dan Goodin writes:

> *For almost two years, Microsoft officials botched a key Windows defense, an unexplained lapse that left customers open to a malware infection technique that has been especially effective in recent months.*
>
> *Microsoft officials have steadfastly asserted that Windows Update will automatically add new software drivers to a blocklist designed to thwart a well-known trick in the malware infection playbook. The malware technique—known as BYOVD, short for "bring your own vulnerable driver"—makes it easy for an attacker with administrative control to bypass Windows kernel protections. Rather than writing an exploit from scratch, the attacker simply installs any one of dozens of third-party drivers with known vulnerabilities. Then the attacker exploits those vulnerabilities to gain instant access to some of the most fortified regions of Windows.*
>
> *It turns out, however, that Windows was not properly downloading and applying updates to the driver blocklist, leaving users vulnerable to new BYOVD attacks.*

Imagine that! How could that possibly happen? Raise your hand if you're surprised. Dan fleshes this out with some nice background, writing ...

BYOVD has been a fact of life for at least a decade. Malware dubbed "Slingshot" employed BYOVD since at least 2012, and other early entrants to the BYOVD scene included LoJax, InvisiMole, and RobbinHood.

Over the past couple of years, we have seen a rash of new BYOVD attacks. One such attack late last year was carried out by the North Korean government-backed Lazarus group. It used a decommissioned Dell driver with a high-severity vulnerability to target an employee of an aerospace company in the Netherlands and a political journalist in Belgium.

In a separate BYOVD attack a few months ago, cybercriminals installed the BlackByte ransomware by installing and then exploiting a buggy driver for Micro-Star's MSI AfterBurner 4.6.2.15658, a widely used graphics card overclocking utility.

In July, a ransomware threat group installed the driver mhyprot2.sys—a deprecated anti-cheat driver used by the wildly popular game Genshin Impact—during targeted attacks that went on to exploit a code execution vulnerability in the driver to burrow further into Windows.

A month earlier, criminals spreading the AvosLocker ransomware likewise abused the vulnerable Avast anti-rootkit driver aswarpot.sys to bypass virus scanning.

Entire blog posts have been devoted to enumerating the growing instances of BYOVD attacks, with posts from security firm Eclypsium and ESET among the most notable.

In other words, these are very real threats and attacks which could be, and should be, thwarted by Windows, but are not being. Eclypsium's blog post is cleverly titled *"Screwed Drivers – Signed, Sealed, Delivered"* and ESET's is titled: *"Signed kernel drivers – Unguarded gateway to Windows' core"* and Eclypsium enumerates the publishers of such drivers, which include: ASRock, ASUSTeK, ATI/AMD, Biostar, EVGA, Getac, GIGABYTE, Huawei, Insyde, Intel, MSI, NVIDIA, Phoenix Technologies, Realtek Semi, SuperMicro and Toshiba. All good companies and all capable of making mistakes.

Dan writes:

Microsoft is acutely aware of the BYOVD threat and has been working on defenses to stop these attacks, mainly by creating mechanisms to stop Windows from loading signed-but-vulnerable drivers. The most common mechanism for driver blocking uses a combination of what's called memory integrity and HVCI, short for Hypervisor-Protected Code Integrity. A separate mechanism for preventing bad drivers from being written to disk is known as ASR, or Attack Surface Reduction.

Unfortunately, neither approach seems to have worked as well as intended.

As we'll see, that statement of Dan's is actually being quite kind. He continues…

Microsoft has touted these protections since at least March 2020, when the company published a post promoting "Secured Core" PCs, which have HVCI enabled right out of the box. Microsoft presented Secured Core PCs (and HVCI in general) as a panacea for in-the-wild BYOVD attacks, stemming either from buggy drivers or "wormhole" drivers (those which are vulnerable by design). The company wrote:

"In our research, we identified over 50 vendors that have published many such wormhole drivers. We actively work with these vendors and determine an action plan to remediate these drivers. In order to further help customers identify these drivers and take necessary measures, we built an automated way in which we can block vulnerable drivers and that is updated through Windows update. Customers can also manage their own blocklist as outlined in the sections below."

The post went on to say that "Microsoft threat research teams continuously monitor the threat ecosystem and update the list of drivers that [are] in the Microsoft-supplied blocklist. This blocklist is pushed down to devices via Windows update."

A few months later, Microsoft Senior VP of Enterprise and OS Security David Weston tweeted that by turning on these protections, Windows users were safe from an ongoing BYOVD attack that had recently made the rounds.

Weston wrote: "Security vendors are going to tell you [that you] need to buy their stuff, but Windows has everything you need to block it."

Multiple Microsoft posts have made the same claim about automatic updating ever since. One from last December said that signed drivers reported to be vulnerable are blocked by default through Microsoft's automated Windows Update mechanism when Windows 10 has HVCI enabled.

But as I was reporting on the North Korean attacks mentioned above, I wanted to make sure this heavily promoted driver-blocking feature was working as advertised on my Windows 10 machine. Yes, I had memory integrity turned on in Windows Security > Device security > Core isolation, but I saw no evidence that a list of banned drivers was periodically updated.

So I reached out to Microsoft and asked if someone would provide me with background about how the protection worked. The response: Microsoft had "nothing to share" with me.

I then turned to Peter Kálnai, a researcher at security firm ESET who has had plenty to share about BYOVD attacks. I asked him for help testing this driver blocklist feature. Very quickly, we found it lacking. When Kálnai enabled HVCI on a Windows 10 Enterprise system in his lab, for instance, the machine loaded the vulnerable Dell driver that had recently been exploited by Lazarus.

Around the same time, researchers, including Will Dormann, a senior vulnerability analyst at security firm ANALYGENCE, had been tweeting for weeks that various drivers known to be actively used in BYOVD attacks were not being blocked the way Microsoft advertised.

One Dormann observation was that, even with HVCI turned on, his lab machines loaded a vulnerable driver known as WinRing0 just fine.

Upon further investigation, Dormann discovered that this vulnerable WinRing0 driver wasn't present in the Microsoft-recommended driver block rules. In the same thread, he went on to show that, despite Microsoft's claims that ASR is capable of blocking vulnerable drivers from being written to disk, he could find no evidence that this feature worked at all. Microsoft has yet to address this criticism or to provide Dormann with guidance.

Dormann went on to discover that the driver blocklist for HVCI-enabled Windows 10 machines hadn't been updated since 2019, and the initial blocklist for Server 2019 only included two drivers.

As scrutiny of this situation increased, a Microsoft project manager finally admitted that something had gone wrong with the update process for the driver blocklist. The manager tweeted that Microsoft was "fixing the issues with our servicing process which has prevented devices from receiving updates to the policy."

What the program manager was saying boiled down to this: If you thought HVCI was protecting you from recent BYOVD attacks, you were probably wrong. Windows 10 hadn't updated the list in almost three years.

Coinciding with the project manager's tweet, Microsoft released a tool that allowed Windows 10 users to deploy the blocklist updates that had been held back for three years. But this is a one-time update process; it is not yet clear if Microsoft can or will push automatic updates to the driver blocklist through Windows Update.

https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-driver-block-rules

While Microsoft's response to my questions about driver blocklist updating was indifference, company employees have been actively dismissive to admins and researchers who began asking their own questions about the topic. Recently, for instance, when Dormann pointed out a demonstrably false claim in an August tweet from Weston—that ASR ensured that updated driver blocking happened automatically—Weston did not apologize or even admit the problems. Rather than confirming an update lapse that spanned more than two years, Weston bristled, saying only that updates "are in the servicing pipeline" and that Microsoft has already "provided a tool to do it right now."

The closest Microsoft has come to an admission of failure is a comment from a company representative, saying, "The vulnerable driver list is regularly updated, however we received feedback there has been a gap in synchronization across OS versions. We have corrected this and it will be serviced in upcoming and future Windows Updates. The documentation page will be updated as new updates are released."

The representative didn't say how long the gap lasted or what upcoming and future Windows updates would fully fix the problem.

Dan's coverage of this mess continues with some powershell scripting that allows individuals to take matters into their own hands. I've placed a link to his entire ArsTechnica article in the show notes. But everyone gets the idea by now. We have another example, not only of gross incompetence, but also of denial and belligerence when employees are presented with embarrassing truths.

https://arstechnica.com/information-technology/2022/10/how-a-microsoft-blunder-opened-millions-of-pcs-to-potent-malware-attacks/

The security industry has been blogging about this and posting about this and waving their arms in the air trying to get Microsoft's attention. Perhaps now that a high profile report has been pulled together thanks to Dan's reporting the pressure will have escalated enough to get this fixed.

Each week, when I listen to Paul and Mary Jo on Windows Weekly, they are every bit as puzzled by the decisions that Microsoft is making. I sincerely hope that this is a pendulum swing, that we are at the nadir now, and that things are going to begin swinging back in the right direction. I'm still not ready to give up on Windows — it's the best user experience in the world — and I know that the enterprise world has no other choice — it's Windows. Unfortunately, Microsoft knows that too. Let's hope that they can and will start getting their act together. I always say that anyone can make a mistake. But this feels like bad policy from on high.

Okay, what else happened this past week??

Microsoft has finally added an RSS feed for Windows Updates!
Yes, after years of pleading from its customers, Microsoft has finally made available an RSS feed for its security updates portal: https://api.msrc.microsoft.com/update-guide/rss

https://msrc-blog.microsoft.com/2022/10/12/14921/

I have a link to the announcement in the show notes and to the RSS feed. I'm sure that this will be of interest to many of our listeners.

Passkeys [dot] Dev
"Passkeys", the industry's first agreed upon replacement for passwords now has its own useful promotional website at: https://passkeys.dev/. The homepage has that original Google/Microsoft passkeys demo video that we saw at the passkey's launch, but the most interesting content for most of us who have been following along is probably a list, though still quite short, of websites where the Passkeys logon experience can be explored. Under "Docs" > "Tools & Libraries" > "Test & Demo Sites" we find WebAuthn.io and passkeys.io. Also, Yubico has a demo site and WebAuthn.me appears to be available. I don't know whether this list is exhaustive. I would

expect that Apple, Google and Microsoft would be supporting their own Passkeys standards soon. There's also a "Device Support" page which contains a nice grid of which versions or what supports what level of Passkeys. So this will be a site to keep an eye on.

Largest DDoS attack

Cloudflare's quarterly DDoS threat report for the just ended 3rd quarter of 2022, noted that it had mitigated a large-scale DDoS attack that reached an astonishing 2.5 Terabits per second — or restated, two thousand, five hundred gigabits per second. The attack was launched by a Mirai botnet variant and aimed at the Wynncraft Minecraft service. I was curious, so I went over to the Wynncraft site: https://wynncraft.com/. It's kinda cool looking.

Signal will be dropping its SMS/MMS support

Signal said that it will be dropping its long-standing fallback support for sending and receiving SMS and MMS messages in its Android app in order to improve user privacy and security. As it was, SMS and MMS were only supported under Android and they were a remnant from the earliest days of the Signal service when it was known as "TextSecure" – remember that?

Brute-force protection for Windows local admin accounts

Oh! There was some good news for Windows users. We previously talked about Windows Remote Desktop Protocol (RDP) finally receiving some relief in the form of failed authentication attempt rate limiting. With last Tuesday's October updates, Windows 10 and 11 (and Windows 7 and Server 2008 R2 on extended service plans) will have received new Group Policy features which enable the implementation of similar lockout policies for LOCAL administrative account logins.

Since I'm sure that this will be of interest to our listeners I'll share some details.
Microsoft's posting explained:

In an effort to prevent further brute force attacks/attempts, we are implementing account lockouts for Administrator accounts. Beginning with the October 11, 2022 or later Windows cumulative updates, a local policy will be available to enable local administrator account lockouts. This policy can be found under Local Computer Policy\Computer Configuration\Windows Settings\Security Settings\Account Policies\Account Lockout Policies.

For existing machines, setting this value to Enabled on existing machines using a local or domain GPO will enable the ability to lock out Administrator accounts. Such environments should also consider setting the other three policies under Account Lockout Policies; our baseline recommendation is to set them to 10/10/10. This means an account would be locked out after 10 failed attempts within 10 minutes and the lockout would last for 10 minutes, after which the account would be unlocked automatically.

For new machines on Windows 11, version 22H2, or any new machines that include the October 11, 2022 Windows cumulative updates before the initial setup,these settings will be set by default at system setup. This occurs when the SAM database is first instantiated on a new machine. So, if a new machine was set up and then had the October updates installed later, it will not be secure by default and will require the policy settings above. If you do not

want these policies to apply to your new computer, you can set the local policy above or create a group policy to apply the Disabled setting for "Allow Administrator account lockout."

Additionally, we are now enforcing password complexity on new machines if a local administrator account is used. The password must have at least three of the four basic character types (lower case, upper case, numbers, and symbols). This will help further protect these accounts from being compromised because of a brute force attack. However, if you want to use a less complex password, you can still set the appropriate password policies in Local Computer Policy\Computer Configuration\Windows Settings\Security Settings\Account Policies\Password Policy.

It's also noteworthy that other reporting has indicated that a similar feature to block SMB-based brute-force attacks is in the works.

Other than that...

- In the past week there were more DeFi and cryptocurrency bridge exploits and currency rip-offs. No surprise

- A nasty Fortinet high-end enterprise security appliance 0-day – which was confirmed last week and has been under active use in attacks – has now been fully elucidated in an unauthorized and arguably premature public disclosure. It's expected that attacks will soon escalate. When Fortinet first learned of the vulnerability privately several weeks ago, they quietly updated their code and sent private messages to their customers urging them to update immediately because a serious authentication bypass had been discovered. Breaking from industry standard protocol, Fortinet chose not to go public at the time.

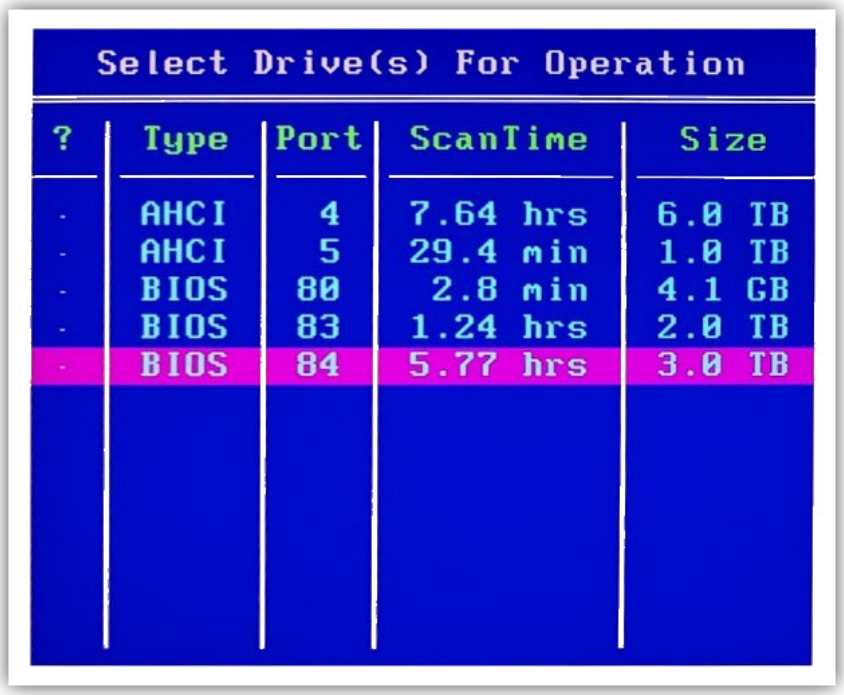
- A week ago, last Tuesday, the White House put out a press release saying that it's working on a cybersecurity label that would be applied to smart (IoT) devices, akin to the Underwriter's Labs UL seal of approval, to help inform Americans which devices "meet the highest cybersecurity standards to protect against hacking and other cyber vulnerabilities." The Administration said it plans to meet with vendors, industry groups, and government agencies later this month to discuss how this new labeling scheme should be managed. The White House said the new cybersecurity labels will first be mandated for "the most common, and often most at-risk, technologies — routers and home cameras — to deliver the most impact, most quickly."

  It'll be interesting to see how this develops. What will be the requirements imposed upon devices to receive these cybersecurity approval labels? Are they going to be worthless, or worthwhile? And how will their application be enforced?

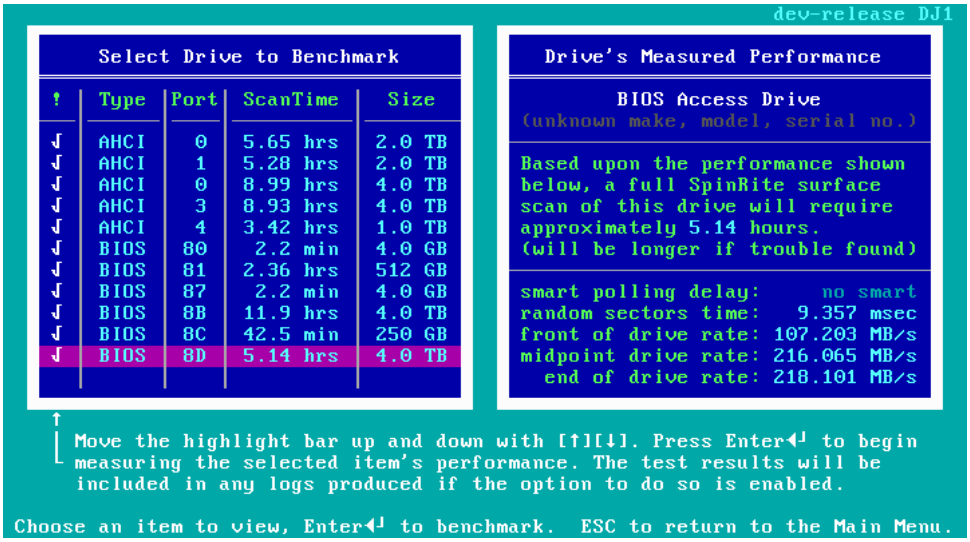
- The US Treasury's Financial Crimes Enforcement Network (FinCEN) fined the cryptocurrency platform Bittrex $29.2 million for failing to detect and block payments to sanctioned entities and also failing to detect payments to dark web markets and ransomware groups. FinCEN said Bittrex made over 116,000 transactions valued at over $260 million to sanctioned entities and connected to criminal activity over the past few years. Apparently, as few as two minimally trained employees were tasked with monitoring more than 20,000 individual transactions per day. In other words, Bittrex wasn't taking its monitoring obligations seriously.

SpinRite

Two more of SpinRite's 378 registered development testers weighed in since last week's podcast with some interesting performance numbers that I wanted to share. The first screen shows a 6.0 terabyte drive, attached to his machine's SATA port #4, having a full drive read-scan time of only 7.64 hours. That's not quite one terabyte per hour, but it's very close:



The other recent report shows a similar level of performance through USB. This guy has 11 drives attached to his machine, and the one that's highlighted at the bottom is an external 4 terabyte drive that SpinRite 6.1 will be able to completely read-scan in 5.14 hours. We've never seen that sort of performance until now. It makes SpinRite practical once again on huge drives:



Until we get to SpinRite 7.1, which will add native hardware USB drivers, SpinRite's maximum USB performance will utterly depend upon the machine's BIOS. But it **can** be very fast.

In reviewing the past week's worth of Twitter communications I encountered a SpinRite success story that I wanted to share:

> ***Ryan Becker / @rb14060***
> *Steve, wanted to share a SpinRite success story. A friend of mine called me saying his 4 year old laptop was running incredibly slow to the point of him not being able to do any work. Upon my arrival I found that task manager was reporting 100% disk utilization with nothing running in the background. Immediately suspecting a failing drive, I recommended he purchase an SSD and have me clone the drive over. However, both CloneZilla and Macrium Reflect gave data read errors partway through the drive and aborted the clone. He is an insurance agent with hundreds of client files, and of course, no backup. So I dug up my copy of SpinRite and loaded it to a USB stick, only to have the laptop fail to recognize that the stick was bootable. Luckily, InitDisk with the FreeDOS option made the laptop recognize the stick, and I was able to copy the SpinRite EXE to the drive. I set SpinRite off in level 2 and it took the better part of the day to run. After it completed, I attempted to clone the drive again in Macrium and this time total success. He is now happily chugging away with an SSD and all of his data is intact. Thanks for a wonderful product. Looking forward to 6.1 and beyond. Please feel free to share this on the podcast if you wish.*

Well, first of all, Ryan, thanks for sharing your success. It's really going to be fun to be fielding a bunch more of those once SpinRite 6.1 is in everyone's hands. And note that the first failure of SpinRite 6.0 to boot is the reason I took the time, before doing anything else, to create InitDisk. SpinRite 6.1 will still be able to create a bootable diskette, but only because all of that code is already written, and it's actually very nice. But it's clear that the future is bootable USB thumb drives, so v6.1 will, of course, incorporate InitDisk's quite robust USB boot setup technology.

Closing The Loop

Guillermo García / @gmogarciag

> *Hi. Listener since episode 001. On SN892 you and Leo discuss the benefits of using uBlock to filter cookie pop-ups. As I am very much aware of tracking, I take the time to configure each one to deny all, if possible. I wonder on which state will the cookies be set when the pop-up is blocked. Maybe they remain all on. Any insights??*

Timo Grün / @Khoji

> *RE SPINRITE: Great to hear about all the exciting developments. Kind of frustrating to know that it can still only be used on old computers in my attic.*

We currently have 378 SpinRite development testers registered in our GitLab instance. So, many people have machines that will run SpinRite on machines that still offer a BIOS. But I'm 100% sympathetic to the need for SpinRite to boot over UEFI. The BIOS is the past and UEFI is today and tomorrow. And that's the reason for my changing plans and deferring native support for USB and NVMe until **after** SpinRite is able to boot and run on a UEFI-only machine without any BIOS and without DOS. So I will be getting there as quickly as I can. :)

Miscellany

xchg rax, rax and "xorpd":

Initially I wanted to publish the book through a publisher, but no publisher wanted to go with the "minimalistic" design. One publisher said he could publish the book if I added explanations for every code snippet, but I wasn't willing to do so.

Another friend recommended adding QR codes to the pages, linking to explanations, but I wasn't willing to do this either. I remember trying to decipher each of those code snippets myself, and I felt that I could not let my readers down by handing them an easy solution.

https://esoteric.codes/blog/xchg-rax-rax:

Much of the joy of the book comes from discovering the nuances of these tiny programs; many rely on assembly-specific tricks that do not really translate "up" into the virtual machines defined by higher-level programming languages.

Those who have trouble deciphering can view xorpd's video series Assembly Language Adventures, which teaches assembly programming beginning with the very basics, reaching a level of expertise through 29 hours of instruction. Xorpd created the book as a companion piece while working on the video, pulling his favorite assembly snippets together.

Assembly Language Adventures: https://www.xorpd.net/pages/x86_adventures.html

What is Assembly Language?
A computer only knows how to execute a small set of commands, or instructions. Those are really simple commands, such as adding or subtracting numbers, comparing numbers and so on. Assembly language is the language of those commands. Using Assembly language you can create computer programs that instruct a computer to do things in the most basic level possible.

Why learn x86 Assembly Language?

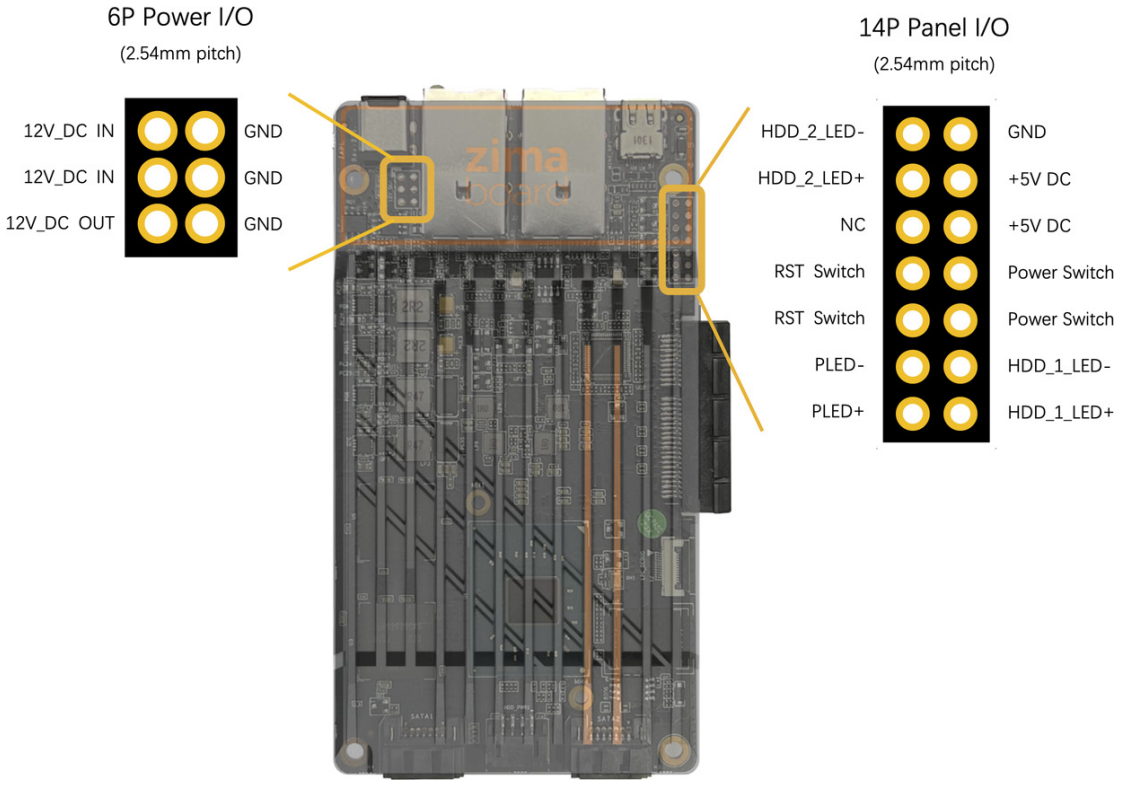
- You are the kind of person who really likes to know how things work. In this course you are going to get solid understanding on how computer programs work from the inside.
- Become a better programmer - Knowing how things work down there will help you take better decisions, even as a high level programmer. If you were always wondering what is the stack, or what are those pointers everyone talks about, you came to the right place.
- Write faster code - When you really want to get the most of your processor, writing in raw Assembly is needed. We are not going to talk about optimizations in this course, however you will get a solid foundations so that you can continue exploring on your own.
- You want to become a reverse engineer or a security researcher, read the code of viruses or look for software vulnerabilities. As most of the time the original source code will not be available to you, solid understanding of x86 Assembly Language is mandatory.

ZimaBoard Goodness

I've spent all of the last week deep into working on SpinRite and that work has been exclusively with the $120 ZimaBoard single board x86 computer that I talked about last week. I love it. One problem I had was that when I'm doing things at the machine level in the debugger, I might manually move the program counter somewhere out of sequence to force some piece of code to run. It's easy to do and super-useful, but that can leave the machine in an unstable state if I attempt to proceed or exit. The bottom line is that the machine is often hanging hard so that the keyboard's famous three fingered salute (CTRL-ALT-DEL) does nothing. And unlike most (but not all) desktop machines, because it's meant to be used as an embedded appliance, it doesn't have a hardware reset button. So whenever the board was hanging I had been forced to pull the power, wait a second, and plug it back in. I hate doing that. And since my current test drive is a 2 terabyte Seagate, it's being forced to spin up and down, which I also hate.

But, when I had disassembled the ZimaBoard earlier—of course I took it apart, that's a required part of the process of getting to know it and falling in love with it—I noticed 14 printed circuit pads arranged in a 2x7 grid at the card's edge just in front of the external PCIe connector. And the case had a cutout there which allowed those 14 pins to be accessed externally. So I was hoping that among those pins might be a signal ground and a hardware reset line to which I could attach a hardware reset button.

So, over the weekend I decided to seek the help of ZimaBoard's creators. I found that they maintain a very active community on Discord, so I jumped online and posted my question in the evening to their support channel. The next morning, when I checked, not only was there an answer, but my prayers were answered. It turns out that the 14-pin pads are a complete PC front panel extension. There's a power button, a reset button & connections for 3 activity LEDs:



https://docs.zimaboard.com/docs/Hardware-Interface-Introduction.html#Pinouts

Password Change Automation

We should begin with a brief refresher about so-called “well-known” website assets.

The most famous of these is the venerable "/robots.txt" file. When automated spiders or 'bots began exploring the web, and as websites began evolving beyond a collection of static web pages, it started becoming possible for bots to get stuck in infinite loops at a site, following a link that led to another that led back to the first one, and so forth. Or they might begin rapidly requesting all of a site's dynamically generated web pages to place an undue load on the site’s web server. So a convention was created. When a bot would enter a site, it would check for a specific file named "/robots.txt" residing in the root directory of the site. If present, that file would provide 'bots with a series of hints, essentially some guidance metadata, about where they could and could not safely venture. For example, my GRC.COM website has a robots.txt file containing:

```
User-agent: *
Disallow: /x
Disallow: /ppp
Disallow: /ppp?
Disallow: /cookies/forensics.htm
```

There’s a User-agent specification which might be an asterisk as it is in my site’s case, or it might single out one or more user-agents (bots, spiders, search engines) by name. Then in my case I have a Disallow for “/x” which is an alias for GRC’s scripts directly where it makes no sense for any search engine to wander. In some cases I wanted friendlier URLs without the /x, so I’m also asking bots to stay away from any URL beginning with /PPP since those are GRC’s Perfect Paper Passwords pages. And I also ask them to keep out of the /cookies/forensics.htm page since that’s really only of any use for examining a browser and its cookie management.

And TWiT.tv's /robots.txt file looks like this:

```
#  88888888888 888       888 d8b 88888888888  888
#      888     888   o   888 Y8P     888      888
#      888     888  d8b  888         888      888
#      888     888 d888b 888 888     888      888888 888  888
#      888     888d88888b888 888     888      888    888  888
#      888     88888P Y88888 888     888      888    Y88  88P
#      888     8888P   Y8888 888     888  d8b Y88b.   Y8bd8P
#      888     888P     Y888 888     888  Y8P  "Y888   Y88P
#

User-agent: *
Crawl-delay: 10
Sitemap: https://twit.tv/sitemap.xml
```

TWiT's file specifies a "Crawl-delay" of "10" which asks spiders, bots and search engines to only pull one page every ten seconds. And the "Sitemap" entry points to an XML format sitemap so that search engines will be able to discover everything that TWiT wants to have indexed.

Leo and I both have comparatively simple sites. Amazon.com's /robots.txt file is 152 lines of mostly "Disallow" URLs. Facebook weighs in at a hefty 610 lines and it's sort of entertaining to browse through. It begins with an off putting block of text:

```
# Notice: Collection of data on Facebook through automated means is
# prohibited unless you have express written permission from Facebook
# and may only be conducted for the limited purpose contained in said
# permission.
# See: http://www.facebook.com/apps/site_scraping_tos_terms.php
```

Then it goes on to list, by bot, where they are permitted to venture. We have the Applebot, the baiduspider, Bingbot (ya gotta love just saying "Bingbot"), Discordbot, something known as facebookexternalhit, Googlebot, Googlebot-Image, ia_archiver, LinkedInBot, msnbot, Naverbot, Pinterestbot, then we have the, I kid you not, "Screaming Frog SEO Spider", seznambot, Slurp, teoma, TelegramBot, Twitterbot, Yandex, Yeti. Those were all "Disallow" URLs. Then the file goes through the entire list again giving them specific "Allow" URLs.

So there are many bots roaming the Internet these days. Of course, there's no practical way for any website to refuse to serve pages to any agent that wishes to request them. In other words, there's no enforcement mechanism; the whole \robots.txt facility is simply advisory.

When it was recognized that beyond a single "robots.txt" file there were a great many more sorts of metadata that websites might wish to publish, not to users but to automated visiting bots, scanners and other tools, it became clear that a more mature mechanism was needed. And the first order of business was to avoid cluttering up the site's root directory with a growing number of random metadata files. So the World Wide Web Consortium—the W3C—standardized upon the placement of everything in a specially designated subdirectory of the root. That directory is named .well-known, or /.well-known/.

Wikipedia explains it this way:

"A well-known URI is a Uniform Resource Identifier for URL path prefixes that start with /.well-known/. They are implemented in web servers so that requests to the servers for well-known services or information are available at URLs consistent well-known locations across servers."

Well-known URIs are Uniform Resource Identifiers defined by the IETF in RFC 8615.They are URL path prefixes that start with /.well-known/. This implementation is in response to the common expectation for web-based protocols to require certain services or information be available at URLs consistent across servers, regardless of the way URL paths are organized on a particular host. The URIs implemented in web servers so that requests to the servers for well-known services or information are available at URLs consistent with well-known locations across servers.

The IETF has defined a simple way for web servers to hold metadata that any user agent (e.g., web browser) can request. The metadata is useful for various tasks, including directing a web user to use a mobile app instead of the website or indicating the different ways that the site can be secured. The well-known locations are used by web servers to share metadata with user agents; sometimes these are files and sometimes these are requests for information from the web server software itself. The way to declare the different metadata requests that can be provided is standardized by the IETF so that other developers know how to find and use this information.

The Wikipedia article lists 46 different items, including the one we'll be talking about in a minute. Most of the 46 well-known item names are obscure, but a few are interesting.

There's "keybase.txt" which Wikipedia says is *"Used by the Keybase project to identify a proof that one or more people whose public keys may be retrieved using the Keybase service have administrative control over the origin server from which it is retrieved."*

The one we originally talked about when we first introduced the concept of the /.well-known/ subdirectory was "security.txt". Wikipedia reminds us that *"security.txt is a proposed standard for websites' security information that is meant to allow security researchers to easily report security vulnerabilities. The standard prescribes a text file called "security.txt" in the well known location, similar in syntax to robots.txt but intended to be machine- and human-readable, for those wishing to contact a website's owner about security issues. security.txt files have been adopted by Google, GitHub, LinkedIn, and Facebook."*

As we know, security researchers have often been frustrated by the difficulty in finding the person who should receive security problem reports. They'll send an urgent eMail to the "Contact Us" info and either never receive any reply or receive a canned *"Thanks for contacting us, your query will be examined and the proper person will get back to you shortly."* reply. So the idea behind "security.txt" located in the /.well-known/ directory, is to allow a site's technical support staff (likely not upper management) to prearrange a means for being directly informed of things they **want** to know that someone might discover.

Two engineers at Apple, Ricky Mondello and Theresa O'Connor, realized that an addition to the existing ".well-known" facility could be employed to help users — and perhaps their password managers and other tools — know where to go to change their passwords for ANY supporting site. It's a small thing, but some of the best ideas are.

As we know, the problem is that there is no commonality among websites for logging in and out, and managing one's identity; it's a completely ad hoc invention for each website. We're beginning to see some coalescence of UI features, with the idea of account management being located in the upper right corner of website pages. That's becoming increasingly common. But what is definitely lacking is any generic direct access mechanism which would allow a user, or some automation, to get to specific aspects of a site's account management. In every case, it's currently necessary to click on a series of links, looking at the result of each click, make a best guess as to what to click next, as we navigate to a desired account management feature.

The idea that occurred to Ricky and Theresa was to add an object to the /.well-known/ collection named “change-password” (“change” hyphen “password”). Whenever that resource was requested, the reply would be a URL which the requester should then follow in order to be immediately presented with a site's password changing page.

Once this has caught on, you can imagine that password managers and web browsers would add a “change site password” feature to their own user interfaces. When their user visits a site, just as browsers currently request the favicon to show the site's small icon identity, they would passively query for the presence of the site's “change-password” resource in the /.well-known/ subdirectory. If the query returns a “404 Not Found” then the browser's or password manager's “change site password” option would be disabled and grayed-out. But if the query returns a change password URL, the client's UI feature would be enabled. And if its user should click on the “change site password” feature, they would be immediately jumped to the proper page at that site having short-circuited any and all intermediate steps to get there.

It's true that some password managers have taken it upon themselves to offer similar features across a limited and specific collection of sites. But this has been accomplished through brute force automation of the user-facing user interface for each specific site, which is prone to failure if or when a site upgrades or updates its users' experience. What this new “change-password” standard accomplishes is to provide a means for cutting out all intervening guesswork and intermediate stages to provide a URL that will take its visitor directly to that page.

And I mentioned that it's a standard, because it is: The W3C has taken this up and has published the first draft of this new addition to the /.well-known/ website metadata:

https://www.w3.org/TR/2022/WD-change-password-url-20220927/

The W3C titled their specification page: **“A Well-Known URL for Changing Passwords”**

... Which is somewhat unfortunate since some of the tech press has apparently only read the title and assumed that this is more than it is. As we've seen, this doesn't actually change anything. It simply redirects its visitor to the website's password change page, in the process, transparently bypassing all of the intervening steps. In other words, this is a “baby step”.

We could wish that the URL returned was to an XML-format SOAP-style API endpoint which would entirely automate the process of authenticating the user with their current username and password, accept the replacement password, and confirm that this update has been made at the server side. But that's not what we're getting.

But are getting the first small baby steps in that direction. It has the benefit that it should be quite easy to implement, and any common frameworks should be able to easily support it so that it can become widespread. Baby steps.