SECURITY NOW!

Transcript of Episode #893

Password Change Automation

Description: 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 U.S. 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.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-893.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-893-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Man, Microsoft has really blown it this time. Take a look at why you should never use the Office encryption technology, and why you maybe shouldn't trust their device driver security. Oh, boy. We'll also talk about Cloudflare and the largest DDoS attack ever, the White House's proposal on an IoT cybersecurity seal of approval, and then we'll about proposals for a better way to change all your passwords. It's that and a whole lot more, coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 893, recorded Tuesday, October 18th, 2022: Password Change Automation.

It's time for Security Now!. Oh, boy. I look forward to this all week long because I just kind of, you know, every time I see a story about security, ransomware, viruses, I say, "I wonder what Steve will say?" And now we know. Steve Gibson is here of GRC.com. Hello, sir.

Steve Gibson: Leo, great to be with you for Episode 893.

Leo: Wow.

Steve: Yup. We're getting the hang of this maybe. So this week's topic is Password Change Automation, which is an overstatement of what we're actually getting. But as I explain here, before we wrap up today's podcast, it does represent a baby step in the right direction, and a potentially exciting new standard. So we're going to talk about that. But first we're going to look at several more serious Microsoft security failures which have just come to light, and a new, useful Windows security feature that was just added with Patch Tuesday's, last Tuesday's updates.

We also have a new Passkeys, well, I'm sorry. The new Passkeys logon technology has received its own website which we can use to monitor Passkeys' progress moving forward. Cloudflare logs another record-breaking DDoS attack, just insane levels of traffic, which if you didn't have one of the main DDoS mitigators in front of your server, just even pouring ice on it wouldn't help. It would just melt down. We've got the news of Signal, one of the premier end-to-end encryption messaging platforms, dropping their legacy support, I didn't even know they still had it, of SMS and MMS on the Android platform.

Leo: Yeah. You don't know because you use an iPhone.

Steve: Right.

Leo: But on the Android it's a big deal because then I can use it as my sole messenger.

Steve: Right, right.

Leo: Which I won't be able to going forward.

Steve: Nope. Also the company Fortinet has sort of attempted to keep a new bad authentication bypass quiet. It was a zero-day that was affecting their customers. We'll talk about how that turned out for them. Also the White House proposes work on an IoT cybersecurity seal of approval, and the U.S. Treasury department has - yeah, god help us - has levied a heavy fine against a cryptocurrency exchange for not caring who they sent money to. I also have some updates on my work on SpinRite and on my just-discovered, and I'm more in love with it than ever, ZimaBoard that we talked about last week. I'll share two pieces of listeners' feedback. Then we're going to finish by examining, as I mentioned, this new standardized means of accessing websites' password change pages. Oh, and we have the first-ever Security Now! Video of the Week. Not Picture of the Week because this one needs to be animated and seen to be believed. It is so good.

Leo: Can't wait.

Steve: In fact, I made it our Shortcut of the Week, but not given a number. It's grc.sc/gate for any of our listeners who want to jump ahead. Anyway, lots of fun today.

Leo: Can't wait. It's going to be a great day for Security Now! fans. And we're going to get to that in a moment. But first - what are you laughing about? It's true. It's true.

Steve: Okay. So, yes. That does have sound. And since this is a Twitter video, it defaults to being muted. But the sound is important. For those who are not watching this, we see somebody who is opening a gate.

Leo: And it opens just fine; right?

Steve: Yeah, well, so there's a whole bunch of technology, it's important to recognize. To the left of this is a proximity card reader, a computerized gate mechanism. It's got a solenoid-actuated release.

Leo: And it's got a stalled video. I wish I could play it.

Steve: Solenoid-actuated release that allows the latch to be electronically opened only if this individual's proximity card permits him to go through the gate. So all this technology is in place. And we should explain also that the gate is sort of a wire grid. So it's not a solid gate. You're able to see through it easily. And it's just sort of got a wire screen with, like, gaps of about two inches around it.

Anyway, so this individual is demonstrating to us that, sure enough, the gate is locked. He rattles it, and it won't open. Puts his security card up to the reader. All sorts of technology happens. The ID is read from the card. It's shot off across the planet somewhere to verify that this individual has the credentials required to allow the door to open. Affirmative confirmation returns. The computer processes the message, activates the power on the solenoid, pulls the latch back, and, oh, the gate opens, sure enough.

Now, to demonstrate the problem here, he then closes the gate, attempts to open it again. It will not open. Rattles. But then he notices, wait a minute. On the other side of the gate, through this wire, there's a handle. I wonder what happens if I push that handle down. Whoops. The gate opens.

Leo: Let's try one more time without the sound. I don't know if it's the sound that's screwing it up. But all right. Because it stops here every time with the sound on. Ah, there we go. And bingo, boingo, bongo.

Steve: Yes. Now, this is one of those cell phone videos that's very narrow and very high. So we cannot see to the sides. We're hoping that this gate is not out in the middle of a field by itself.

Leo: It looks like it's a pretty secure facility, given all the technology on that thing.

Steve: Yeah. We have of course seen gates that are out in the middle with well-trodden paths circumventing the gate, making you wonder why anyone even bothered. Anyway - won't open?

Leo: Freezes every time at that point. I don't know why.

Steve: And then he reaches through the gate, pushes down the handle latch, which opens it. So clearly anyone on the other side was meant to be able to come in, or maybe that's probably to leave this protected facility. But you can't enter.

Leo: You can check out, but you just can't check in. No, wait a minute.

Steve: It is just like, yeah, anyway, for anybody who's been listening and wondering what we're talking about, I made this, I gave this a GRC shortcut. So it's grc.sc/gate, G-A-T-E, grc.sc/gate. It's definitely worth checking out.

Leo: And by the way, this guy, I would follow him on Twitter because he has a lot of similar silly things in there.

Steve: Oh.

Leo: Mistakes and so forth in security. So I like his - I might start following his tweet. Here's a picture of a vehicle where there's a cutout in the back, exposing some wires, that says: "When you need to access the CAN bus, but a vendor has installed next-gen AI-driven security controls to thwart attacks." Well, easy enough, you just cut a hole in the fiberglass.

Steve: Oh.

Leo: Oh, lord.

Steve: Yeah, it's very similar. That's great.

Leo: Yup.

Steve: Yeah. So his Twitter name is @HamzahBatha. Again, @HamzahBatha. And as you said, a good feed.

Leo: Yup, yup.

Steve: Okay. So I was torn between titling this piece "Won't Fix" or "Secure Enough." I settled on "Won't Fix," but we'll get to what is secure enough. So "won't fix" is what Microsoft told the guys, actually guy, who noted and reported his or their - he is with a company, a well-known company - their discovery. They said, essentially: "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 once again, which is redundant.

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 that the industry has settled upon currently is the Rijndael

Cipher, which was chosen to be the AES standard. Therefore we also call it the AES Cipher. It's a 128-bit-wide block cipher, meaning that it takes a block of 128 bits at a time, which is 16 bytes, and under the influence of a key, which is typically kept secret, and the key is often 256 bits, it takes those 128 input bits and arranges to map every possible input combination of those 128 bits into a different output combination of 128 bits. That's the encryption.

So as long as the key remains the same, every time the same 128 bits is presented, the same different 128 bits is produced. And of course that's required for the cipher to be useful. Obviously, it must be deterministic and not generate random outputs. Same data in, same data out. 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, the 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 and statistical analysis, significant information can be leaked. And if any of the input text is known, like standard query headers, packet protocol headers or data, boilerplate, or any other overhead which is encrypted as part of this, then someone examining output blocks can see what those known input blocks encrypt to.

So 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. And by now you can probably guess where we're going with this. Someone did use it. Anyway, the most famous and 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. I've got it in the show notes. It's also on Wikipedia's page under "Block Cipher Mode of Operation."

And in the show notes we see three images. On the left is the input, like the so-called plaintext image of the Linux Penguin. The middle is that penguin encrypted using ECB, Electronic Code Book mode, which is to say simply taking the encryption block, encrypting it into a different block.

Leo: How could they even call that encryption?

Steve: Exactly. I know.

Leo: I mean, that's not encryption.

Steve: I know.

Leo: Now, this really is a good example of why ECB is so bad.

Steve: Yes, isn't it? It's just perfect. And what Leo's referring to, for those who aren't seeing this, is you can tell, you can still see that it's the Linux Penguin. I mean, it's not pretty anymore, the colors are lost, but it's like it's not obscured. It's there. Because every time the same block of bits was ciphered, it came out to be the same different block. And in an image context, the image survives. It's not great, but it's there. And compare that to the third frame here, which is just noise. All other modes except ECB result in noise. I mean, like no picture at all, just static. And that's of course what you want from your encryption algorithm.

So what we actually want, as I said, is shown in that rightmost of the three panes, where the result is pure noise without any vestigial remnant of the original image.

Leo: There's no evidence that there was any information; right? That's the key.

Steve: Correct. It is mathematically indistinguishable from entropy, from pure noise

Leo: So there's no information there. Yeah, yeah.

Steve: Yes, yes. And as I said, any cipher mode other than the simplistic ECB results in something like the third image, just noise. Actual encryption which does not leak information about the unencrypted plaintext is what we want. Now, I'm not going to delve into great detail about the other encryption modes since we have carefully and fully covered all this before. And the Wikipedia link that I've got in the show notes 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 popular encryption modes solves this simply by chaining. The most famous of these modes is, and it's as good as any, 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."

So this is from Helsinki, Finland, which is where these guys are located. They said: "Today, WithSecure (formerly known as F-Secure Business) published a security advisory warning organizations of a security flaw in Microsoft Office 365 Message Encryption." So this is Office Message Encryption, or OME for short. So they said: "OME, which is used by organizations to send encrypted emails internally and externally, utilizes the" - wait for it - "Electronic Codebook implementation." And calling it an implementation, even that is kind of a stretch.

They say: "...a mode of operation known to leak certain structural information about messages. Attackers able to obtain OME messages 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." It's the repeated patterns, for example, that allows us to see the Linux Penguin even after it's been "encrypted." And they said: "...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 email archives stolen during a data breach" - notably encrypted email archives encrypted under this ridiculous Electronic Codebook encryption - "or by breaking into someone's email account, email server, or gaining access to backups."

So 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. Yes, because it's not actually encryption. And that the use of BYOK, Bring Your Own Key scheme, does not remedy the problem, meaning it doesn't matter if you privately key this. This is not actually good encryption.

Sintonen shared his research with Microsoft in January of 2022. While Microsoft acknowledged the problem and paid Sintonen via their vulnerability reward program, they opted not to fix the issue. While organizations can mitigate the problem simply by not using the feature, I guess and not assuming that they have encryption, it does not address the risks of adversaries gaining access to existing emails which were previously 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's 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 I'll note that the trouble is not just theoretical. WithSecure's technical write-up placed an image in an Office 365 message, encrypted and sent it using Microsoft's OME (Office Message Encryption), and this was the result. I have another picture in the show notes where you can, as clearly as you could with the Linux Penguin, see the word "Fail" written quite clearly in the image because yes, that's what this is, an encryption fail.

So 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 Office 365 customer. And they know it. So why bother fixing it? Just call it "encrypted" and figure that it's encrypted enough. And this begs the question of how this could have ever happened in the first place. I mean, okay, so, yeah, we're confronted with this, like, horrible design, I mean, I'm reticent to call it a design, you know, I mean, it's like...

Leo: Why does anybody use ECB encryption? I mean, is there some useful reason? Or is it just old?

Steve: No, it's not even old. I mean, nobody ever - the thing that you need with CBC is you need an initialization vector. Remember that I said you take the output of block cipher N and XOR it with the plaintext of block cipher N+1 before encryption. But that then begs the question, okay, how do you start? Because the first encryption won't have a previous result to use. So that's where you need a 128-bit initialization vector. It doesn't need to be secret. It does need to change with every time you use the encryption. But that's simple. Just increment it. Again, it needs to be unique, but it doesn't need to be secret. So you would have to include that with the message, or in a header for the email. But that's trivial, too.

So, I mean, I can't - so, okay. Arguably I know how encryption works. Clearly, whoever designed this doesn't. I mean, Leo, I mean, it's like, whoever, truly, whoever did this doesn't know how to encrypt things. So they just, like, took some symmetric cipher, and I don't even know if it's AES, maybe, hopefully, but they took a symmetric cipher and just said, oh, feed blocks in, send blocks out. Which, as we've seen, isn't good. And galling is that when Microsoft was told this at the beginning of the year, they said yeah, you know, it's encrypted enough. We're going to leave it the way it is.

Leo: It's not encrypted.

Steve: No.

Leo: And it's a lie to say it's encrypted.

Steve: I know.

Leo: It's baffling to me. And why would you even choose this in the first place, let alone not fix it?

Steve: The only thing I can suggest is that this was given to somebody who was utterly incompetent to be given this job. And they said, okay, yeah, done, it's encrypted.

Leo: Might as well just XOR it. You know? Why does ECB exist? Why is it there?

Steve: Because it can. I mean, because, like, just somebody...

Leo: Did somebody once think it was a good way to do it?

Steve: So I think probably it's because once upon a time we had the Caesar cipher, where you took the alphabet, and you skewed it by some number of characters; right?

Leo: Yeah. That's basically what this is; right?

Steve: Yes, exactly. That is Electronic Codebook. The idea is you have something. You look up in a codebook what it maps to, and that's what you write down. And then at the receiving end they have the reverse codebook, where you look up what you got, and you saw what it was mapped to in the first codebook. And so it's sort of there just because it completes the picture. But nobody should use it. I mean, for this reason. And how to do it right is like in the next paragraph. It's like the guy stopped reading Wikipedia after three paragraphs. Just read the fourth paragraph, where it says don't do this. It's like, how does this get into Microsoft Office 365? And then not only does it get in, but they've said, eh, we're going to leave it the way it is. Good enough.

Leo: Unbelievably incompetent.

Steve: Which really does make you worry about what's happening at Microsoft.

Leo: Well, I mean, they could even say, oh, whoops, let's fix it. They're not even going to fix it.

Steve: No.

Leo: Unbelievable. Well, the bad guys must love this.

Steve: Oh, yes.

Leo: Jiminy Christmas.

Steve: Okay. But we're not quite done yet with Microsoft for the week. Ars Technica's coverage of this next piece bore 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."

Leo: This one's also boneheaded.

Steve: Yup. I know, Leo. And, well, we'll see. They even got belligerent toward those who were trying to say, uh, it doesn't work. Anyway, before we get into the details of what was discovered, recall that this is something also that 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. You know, rootkits are an example; right? They're able to filter the API in order to make files disappear from the file system. So they must be absolutely trusted.

But they also contain complex code which requires a level of attention to detail that can sometimes 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 at the time that it was used for signing, it remains valid. But what if a problem is identified later 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 BYOVD, 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 blacklisted. 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 Ars Technica's Dan Goodin writes, he said: "For almost two years" - and later he says three - "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," Dan writes. "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." Probably back when we talked about it. "Malware dubbed 'Slingshot' employed BYOVD since at least 2012, and other early entrants into the BYOVD scene included LoJax, InvisiMole, and RobbinHood. Over the past couple of years we've 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." And note, all of these were known. All should have been blocked. None were. "Entire blog posts," he writes, "have been devoted to enumerating the growing instances of BYOVD attacks, with posts from security firms Eclypsium and ESET among the most notable."

Okay. So 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," writes Dan, "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" - so more than like 2.5 years - "when the company published a post promoting 'Secured Core' PCs, which have HVCI enabled out of the box. Microsoft presented Secured Core PCs, and HVCI in general, as a panacea for in-the-wild BYOVD attacks, stemming from either buggy drivers or 'wormhole' drivers, which are those which were created and vulnerable by design."

Microsoft said: "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 help further 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, writes Dan, "that Microsoft threat research teams continuously monitor the threat ecosystem and update the 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, Microsoft 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 Dan writes: "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" - it's 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 from Microsoft is 'nothing to share.' I then turned to Peter Kalnai, a researcher at security firm ESET, who has had plenty to share about BYOVD attacks. I asked Peter for help testing this driver blocklist feature. Very quickly, we found it lacking. When Kalnai 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 had 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 Windows-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. Gee, imagine that. 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 themselves which had been held back for three years. But this is a one-time update process. It's not yet clear if Microsoft can or will push automatic updates to the driver blocklist through Windows Update." I do have a link in the show notes at this point to the posting that Microsoft made that it will allow end users or hopefully enterprises to update their blocklists.

Dan goes on: "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" - he's a senior VP; right? - "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 that 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 this problem."

So, wow. 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 Ars Technica 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. Obviously Microsoft hasn't bothered to test themselves any of this for the last three years. They said, oh, look what we've got now. But apparently it never worked.

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 to get this fixed. You know, each week 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, and that we are at the nadir now, and that things are going to begin swinging back in the right direction.

I'm still not yet ready to give up on Windows. It's still 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 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. What's good policy for us is taking a break right now, Leo.

Leo: Yes. And we will undoubtedly talk about this tomorrow on Windows Weekly with Paul and Mary Jo. I'd be curious what their take is on both of these stories, frankly.

Steve: And I think, Leo, I'll take over while you take a sip of something.

Leo: So it's my turn. Thank you, Steve. All yours.

Steve: So what else happened this week? Microsoft - wait for it - has finally added an RSS feed for Windows updates. Yes, after endless years of pleading from its customers, Microsoft has finally made available an RSS feed for its security updates portal. I've got the feed link in the show notes.

Leo: That's kind of amazing that they've never had this.

Steve: I know. And like there's been so much pressure on them, it's like, just can we have it as an RSS? And it's like, no, we're going to do it this way. Anyway, they finally said okay. Now, I think Linux is kind of getting the better of them, Leo. They sort of seem to be, you know...

Leo: I'm not saying anything.

Steve: So I've got a link in the show notes. I've got a link to their blog posting announcing it for anybody who's interested. You can find that there.

Passkeys, the industry's first agreed-upon replacement for passwords, now has its own useful promotional website. That's at Passkeys.dev, P-A-S-S-K-E-Y-S dot D-E-V. The home page has that original Google/Microsoft Passkeys demo video that we saw back at Passkeys' launch or announcement. But the most interesting content for most of us who have been following along is probably a list of, though it's still quite short, of websites where the Passkeys logon experience can be explored. So under Docs, Tools & Libraries, Test & Demo Sites, we find WebAuthn.io and Passkeys.io. That's pretty much it for the moment.

Leo: That's kind of disappointing.

Steve: I know, yeah.

Leo: Okay.

Steve: Yubico has got a demo site, and WebAuthn.me appears to be available for testing. I don't know whether this list is exhaustive. I would expect that Apple, Google, and Microsoft at least...

Leo: It's now in iOS 16, and it's going to be in Android. I think...

Steve: Yup.

Leo: I feel like probably this list is not exhaustive because there's probably new guys coming on every five seconds, you know.

Steve: Let's hope.

Leo: Let's hope.

Steve: Although remember it does take some work at the server end. And so again, we'll see how this goes. So hopefully Apple, Google, and Microsoft at least would be supporting their own Passkeys standard soon. There is a device support page which contains a nice grid of which versions of what support what level of Passkeys. So anyway, that'll be something to keep our eyes on.

As I mentioned, Cloudflare's quarterly DDoS threat report for the just-ended third quarter of 2022 noted that it had mitigated a large-scale DDoS attack that reached an astonishing 2.5 tbps, okay; or restated, 2,500 gbps. 2,500 gbps. 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, Wynncraft.com. Never been there before. It's kind of cool-looking to see what a state-of-the-art Minecraft service looks like.

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

Leo: Oh, yeah.

Steve: Yeah, TextSecure. That's what Signal became. Or that's what - wait. Signal became of that. Or something.

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 for what it's worth Windows 7 and Server 2008 R2 if they're on the extended service plans, will have received new Group Policy features which enable the implementation of similar lockout policies for local administrative account logins. That is, you know, somebody banging on your keyboard while you're away at lunch.

Since I'm sure that this will be of interest to our listeners, I'll share some details. Microsoft's posting explained. They said: "In an effort to prevent further brute force attacks/attempts, we are implementing account lockouts for Administrator accounts. Beginning with October 11th, 2022" - which was last week, last Tuesday - "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." That's, you know, under GP Edit.

"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" - well, yeah, you kind of have to, you'll see in a second - "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" - which is just now coming out, as we know - "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." So this becomes the new default for Windows local account login. "This occurs when the SAM database is first instantiated on a new machine." That's the security accounts database. 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."

And finally, they said: "Additionally, we're 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 - lowercase, uppercase, numbers, and symbols. This will help further protect those 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."

So it's also noteworthy that other reporting has indicated that a similar feature to block SMB, that is, Windows Printer and File Sharing-based brute-force attacks, is in the works. So it only took, what, 30 years, and Microsoft is beginning to bring their systems and their network protocols up to a standard that will help protect their users. Yay.

Other than that, in the past week there were more DeFi and cryptocurrency bridge exploits and currency rip-offs. No surprise. As I mentioned, a nasty Fortinet high-end enterprise security appliance zero-day was found. That was confirmed last week and has been under active use in attacks. It's now been, unfortunately, fully elucidated in an unauthorized and arguably premature public disclosure. It's expected that the attacks will soon escalate as a consequence of this.

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 of the seriousness of the authentication bypass that had been discovered. That broke from industry standard protocol of acknowledging an event when it's learned. Fortinet in this case chose not to go public at the time. Well, it's certainly public now, and then some.

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, similar to the Underwriters Labs UL seal of approval, to help inform Americans which devices "meet the highest cybersecurity standards to protect against hacking and other cyber vulnerabilities." Okay. The administration said it plans to meet with vendors, industry groups, and government agencies - that's hopeful - later this month to discuss how this labeling scheme would be managed. The White House said the new cybersecurity labels will first be mandated for "the most common and most at-risk technologies," which they feel are routers and home cameras, "to deliver the most impact, most quickly."

So 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 labels, or worthwhile? And how will their application be enforced? We'll find out.

And finally, the U.S. 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, you know, other financial crimes. 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 very seriously. And now they'll have to pay nearly $30 million in fines. Ouch.

I got a little bit of additional good news on the SpinRite front. Two more of SpinRite's 378 currently 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 TB drive, attached to his machine's SATA port #4, having a full-drive read-scan time of only 7.64 hours. So that's attached to an, you know, and it's AHCI, not USB, AHCI attached to AHCI port 4. So 6TB, able to be scanned by SpinRite in 7.54 hours. So that's not quite a terabyte per hour, it's actually 1.25 hours per terabyte, but it's gratifyingly close. As we can see, SpinRite really, 6.1 really will be able to scream.

The other recent report shows a similar level of performance through USB. This guy has 11 drives attached to his machine. He's a little drive heavy. Six of them are on USB ports. One of them was a boot drive. The one that's highlighted at the bottom is an external 4TB drive that SpinRite 6.1 will be able to completely read-scan in just 5.14 hours. We've never seen that sort of performance until now. As I mentioned before, 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 because that's what we're still going through. But as we see here, it can, given the right BIOS, be very fast. Like no reduction in speed over AHCI at all. I'm sure that was a USB3.0.

And in reviewing the past week's worth of Twitter communications, I encountered a SpinRite success story that I thought would be fun to share. Ryan Becker tweeted from @rb14060. He said: "Steve, wanted to share a SpinRite success story. A friend of mine called me saying his four-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's 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 its 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." That's of course SpinRite 6.0 he's got. "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."

So first of all, Ryan, thanks for sharing your success. It's really going to be fun to be fielding a bunch more of those sorts of stories once 6.1 is in everyone's hands. And note that the first failure of SpinRite 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 kind of cool. But it's clear that the future is bootable USB thumb drives. So 6.1 will of course incorporate InitDisk's quite robust USB boot setup technology. Of course that's why I wrote it in the first place was to be incorporated into all future gizmos that I've got, SpinRite 6 and 7 and beyond and other stuff.

Two pieces of Closing the Loop feedback. Guillermo Garcia said: "Hi. Listener since Episode 001." He said: "On SN-892" - so that was last week - "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 on? Any thoughts?"

Guillermo and everyone, it's interesting. One of the things that I learned caught me by surprise, which was when I was doing that cookie forensics work many years ago, closely looking at the way cookies were being handled by browsers. One of the things the cookie forensics system does is it notes the timestamp on the test cookies which performing those forensics tests set.

What we discovered was that if you, depending upon the browser you were using, if you had cookies enabled before, but then you disabled them, some browsers would stop sending the cookies they already had, whereas some browsers apparently stopped receiving new cookies, but did not block existing cookies. Which I sort of thought was the wrong behavior. If you said "Turn off all cookies," well, I don't want anything to be sent anymore. Some browsers stopped receiving them. So it was a little bit of a glitch which I wondered about at the time. So anyway, that's what would happen if you turned cookies off is, you know, it's worth turning them off and deleting them all from your browser to make sure it's not one of those that keeps sending them.

And Timo Grun, he said: "Regarding 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." Okay. So we currently have 378 development testers registered in our GitLab instance. So many people have machines that will run SpinRite on machines that still only offer a BIOS. But I am 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 any UEFI-only machine without any BIOS and without DOS. So I will be getting there, Timo and everybody else, as quickly as I can.

Leo, I received your books. Thank you.

Leo: Oh, yay.

Steve: Very, very cool.

Leo: You can throw them out. You can do anything you want with them. I just thought they looked like something you might be interested in.

Steve: No, no, no, no. "Hackers Delight" looks like it will be really fun to read through over time. And the book titled "xchg rax,rax," that is really fun. So for people who don't know, it is the plainest, most simple book you could ever have. All it is, you open it, and it is three or four, maybe up to like nine, but sometimes just two, just a few lines of 64-bit Intel assembly code. No explanations. No introduction. That's all they are. And even the page numbers are in hex. I mean, this guy is a hacker's hacker. And I found some quotes from him that I thought would be fun to share. He said, "Initially I wanted to publish the book through a publisher. But no publisher wanted to go with the" - and he has in quotes "minimalistic," which is to say the least - "design." He said: "One publisher said he would publish the book if I added explanations for every code snippet."

Leo: No. No.

Steve: And he said: "But I was not willing to do so." He said: "Another friend recommended adding QR codes to the pages, linking to explanations, but I wasn't willing to do this either."

Leo: Good for him.

Steve: He said: "I remember trying to decipher each of those code snippets myself." So maybe he collected them. He says: "I felt that I could not let my readers down by handing them an easy solution."

Leo: Good.

Steve: So then somebody reviewing the book wrote: "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."

And then he said something interesting: "Those who have trouble deciphering can view xorpd's" - that's the pseudonym of the author of this - "can view xorpd's video series titled '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." So I've got a link in the show notes, xorpd.net/pages/x86_adventures.html, which will take you there.

And there he says: "What is Assembly Language? A computer only knows" - this is him writing. "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."

He says then: "Why learn x86 assembly language?" And he has four bullet points. "You are the kind of person who really likes to know how things work. In this course you're going to get solid understanding on how computer programs work from the inside. Two, become a better programmer. Knowing how things work down there will help you make better decisions, even as a high-level programmer."

Leo: I agree. I agree. Yup.

Steve: Yup. "If you were always wondering what is the 'stack,' or what are those pointers everyone talks about, you came to the right place. Three, write faster code. When you really want to get the most of your processor, writing in raw assembly is needed. We're not going to talk about optimizations in this course. However, you will get a solid foundation so that you can continue exploring on your own. And finally, fourth, 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."

So Leo, again, thank you for the two books. They will absolutely find me spending some time with them. They look great.

Leo: Good.

Steve: And lastly I wanted to share a little bit of additional ZimaBoard goodness. I know that a lot of our listeners loved learning about ZimaBoard last week because I got a lot of tweets and feedback. 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. Or often sometimes it's like to make something run, like to rerun a subroutine where the result wasn't what I expected. So I want to go back to it now, and this time go into it, whereas last time I stepped over it.

So at the machine level in a debugger, this is easy to do and super useful. But it can leave the machine in an unstable state if I attempt to then 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, the ZimaBoard 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 2TB Seagate spinner, it's being forced to spin up and down, which I also hate.

But when I had disassembled the ZimaBoard earlier, upon receiving it, you know, 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, this past 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, I think it was like Saturday evening, to their support channel. The next morning, when I awoke and 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, and connections for three activity LEDs. I've got a picture of that in the show notes for anyone who's interested, and a link to the page that that came from over at docs.zimaboard.com.

So anyway, that's the last thing I needed. When I've been testing SpinRite on all these various motherboards, I've not been mounting them each in a case because there's just no need to. But I have needed a power switch and a reset button. And I found this cool little - it's like a little round hockey puck that's got a big power switch and then a little reset button. I actually use them in reverse. The big easy-to-press button is reset because I'm doing it so often. And the little button powers the machine on and off. And so, and it's just got a - I got it for like, it was a little over $6 from Amazon, that gives me a little portable, you know, an external power switch and reset button, which is really handy. So I've got another one of those that I'll be - actually two that I'll be hooking up to each of my ZimaBoards at my two locations. And now my life is complete.

Leo: Let's talk about changing your passwords automatically.

Steve: Yeah. We're going to 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, like following a link that led to another page, that had a link 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 burden 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, so "/robots.txt." 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. It contains, first of all, it has a statement, User-agent: *, meaning this applies to all who come. Then I have a Disallow: /x. That is an alias for GRC's scripts directory, where it makes no sense for any search engine to wander. In some cases I wanted friendlier URLs without the /x, so I translate those on the fly. But bots would see something else.

So I'm also, for example, asking bots to stay away from any URL beginning with /ppp, since those are GRC's Perfect Paper Passwords pages, where it doesn't make any sense for a bot to go. And I also ask them to stay away from the cookies forensics page because, again, same reason, it's an automated test that doesn't make any sense for a search engine to dip into. And TWiT.tv has a robots.txt file which looks like this. I have it in the show notes. It's got a big TWiT.tv spelled out in block ASCII characters. Then also it has a User-agent: *. Leo's page has a Crawl-delay: 10, which asks spiders, bots, and search engines to only pull one page every 10 seconds, and also offers a sitemap to them which allows them to discover things that they want search engines to see that they might otherwise not find for themselves.

Leo: Yeah, that's supposedly good practice. I don't know.

Steve: Yes, yeah, exactly.

Leo: If you say it's good, I'll keep doing it.

Steve: You know, you and I have comparatively simple sites. Amazon.com's robots.txt file is 152 lines of mostly "Disallow" URLs. Facebook's weighs in at a hefty 610 lines.

Leo: Oh ho ho.

Steve: Yeah.

Leo: We only have three.

Steve: Yeah, exactly. You have three.

Leo: We do have a cool TWiT.tv logo in our robots.txt.

Steve: Yeah, yeah, yeah. Facebook, 610 lines. And it's sort of entertaining to browse through. It begins with an off-putting block of text which reads: "Notice: Collection of data on Facebook through automated means is prohibited" - this sounds like Mark - "unless you have express written permission from Facebook and may only be conducted for the limited purpose contained in said permission." And then there's a - it says "See" and then he's got a URL to site_scraping_tos_terms.

Leo: Holy cow.

Steve: I know. Then it goes on to list, by bot, where they are permitted to venture and where not. We have the Applebot, the Baiduspider, Bingbot - you've got to love just saying "Bingbot" - the Discordbot, something known as facebookexternalhit, also the Googlebot, the Googlebot-Image, ia_archiver, LinkedInBot, MSNbot, Naverbot, Pinterestbot. And then we have, I kid you not, the "Screaming Frog SEO Spider." We've got the SeznamBot, something called Slurp, Teoma, the TelegramBot, the TwitterBot, Yandex, and Yeti. And those were all "Disallow" URLs in a long list. Then the file goes through the entire list again giving them specific "Allow" URLs.

So anyway, obviously 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. But 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, other tools, who knows what, 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 else into a specially designated subdirectory off of the root. That directory is named .well-known, or /.well-known/.

Wikipedia explains it this way. They say: "A well-known URI is a Uniform Resource Identifier for URL path prefixes that start with /.well-known/. They're 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. Well-known URIs are Uniform Resource Identifiers defined by the IETF in RFC 8615." So that's a relatively recent one. They say: "They are URL path prefixes with the start of .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." In other words, let's eliminate website-to-website variations for this one particular purpose.

So Wikipedia says: "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," they write, "has defined a simple way for web servers to hold metadata that any user agent, for example, a 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." Meaning another URL. "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."

Okay. So Wikipedia lists 46 different items, which are currently defined under the .well-known, including the one that we'll be talking about in a minute. So Wikipedia is keeping itself current because this was just released. 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." In other words, it's an authentication mechanism located at .well-known/keybase.txt.

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 describes 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." And doubtless countless others.

So as we know, security researchers have been frustrated in the past by the difficulty in finding the person who should receive problem reports. They'll send an urgent email to the Contact Us info, you know, the only thing they can find, 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," if that ever happens. 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, but the guys who are actually down there pulling wires, to prearrange a means for being directly informed of things that they want to know that someone might discover.

Now, 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, to 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 over and over again for each website. We're beginning to see some coalescence of UI features, you know, 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 on next, as we navigate toward our 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." 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-change 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 the 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 URL password, the client's UI feature would be enabled. And if its user should click on "change site password" feature, they would be immediately jumped to the proper page at that site, having short-circuited any and all intermediate stages to get there, creating complete unification.

It's true that some password managers have taken it upon themselves already to offer somewhat 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 a specific site, which is prone to failure if or when a site upgrades or changes 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 which 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. Its specification page is titled "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 was 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. So it's a "baby step."

We could wish that the URL returned 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 we're 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 quickly. Again, a useful baby step.

Leo: Well, we're making progress, anyway.

Steve: Yup.

Leo: You wouldn't expect a unified way of doing this. I guess maybe there could be, but...

Steve: Oh, yeah. I mean, it wouldn't be difficult for someone to specify that at all.

Leo: Yeah. Someone to specify it, but then somebody else to adopt it in the million pages.

Steve: Yeah. And it really would be cool, when you think about it.

Leo: Oh, it would be, be so huge.

Steve: Well, if Bitwarden, even at this level, if Bitwarden or LastPass, for example, had in their dropdown "Change site password," that jumped you for any site immediately to their password change page, that would be very cool.

Leo: Oh, be great, yeah.

Steve: So this gives us at least that much.

Leo: Good. Yeah. Anything that could automate this process would be very valuable. Would Passkeys help this? Could there be an automated system for that? I guess not because they're no credential to change; right? You don't...

Steve: Right, Passkeys actually, thank god, it obsoletes usernames and passwords completely.

Leo: Right.

Steve: It replaces them both.

Leo: But you must allow, I know you do in SQRL, allow a way for somebody to change their proof; right?

Steve: Oh, sure, yeah, yeah, yeah. You can still have a username and password, and you are able to change your SQRL identity. Actually the way you do it is you give them both your old and your new. So it validates you with the old and then switches over to the new.

Leo: But there is that issue, and I don't know what's going to happen with Passkeys if I lost my phone, you know, there are issues. I mean, this has - there are times you have to change this stuff; right? Even with Passkeys and SQRL.

Steve: Yup. And that's what was so nice about SQRL was if you lost your phone, it didn't matter because your entire identity was a single QR code.

Leo: Right, right.

Steve: And you just put that in a drawer somewhere.

Leo: Steve Gibson. See, that's - I think that's the obvious and smart way to do it, but okay. Fine. Steve is the best. GRC.com's the place to go. You can get a copy of the show there as well as our site, GRC.com. The GRC version actually there's two unique versions. There's a 16Kb audio version, which is, what is that, one seventh, one sixth the size. One sixth the size, I guess. And the advantage of that is you can download it in a limited bandwidth situation. Doesn't sound great, but it's there.

That's the version that Elaine Farris uses because she's, I don't know what, she's got a satellite Internet, I guess, and she's in the middle of horse country. She does the transcriptions, downloads it and types it all out. And that makes it easy to read. It's great for search. It's great to read along as you listen.

Of course Steve has the full quality audio, the 64Kb audio, as well. GRC.com. While you're there, pick up SpinRite, the world's best mass storage and maintenance recovery utility. I was using it the other day, and I saw, oh my god, the copyright 2004. But if you write it right, you don't need to fix it all the time. However, I'm hoping it will be a 2022 copyright, if not, a 2023 copyright for 6.1. That's due imminently. Probably - maybe early 2023. Let's not rush it.

Steve: Yeah. Yeah, it's going to be in full - it'll be running within a week or two, and then in alpha test.

Leo: Right. We've got to make sure it's perfect before we release it. But if you buy now, you'll get a copy of that when that comes out, any day now. We have a copy of the show. We have actually two copies. We have the audio that Steve has, but we also have a video, if for some reason you want to watch. And we do, you know, there are some visual aids. There's stuff to see. There's pictures. That's at TWiT.tv/sn.

There's also, of course, a YouTube channel dedicated to it. That's only video, but a great way to share it with somebody else because YouTube will let you share a little clip, which is nice. And then probably the best way to get it, if you want to listen every week, and I'm sure you do, is subscribe to it in your favorite podcast player. And that way you'll get it automatically the minute we put it out.

We do Security Now! on Tuesdays at about, it was a little late today, sometime between 1:30 and 2:00 p.m. Pacific, 4:30 and 5:00 p.m. Eastern, 20:30 UTC. You can watch us do it live. That's why I mention the times because we actually do stream it as we do it. And that's why the time varies, because shows before go long or whatever. But you can watch the stream at live.twit.tv. You can also chat with us.