SECURITY NOW!

Transcript of Episode #824

Avaddon Ransonomics

Description: This week, believe it or not, we have yet another zero-day stomped out in Chrome. We also have some additional intelligence about the evolution of the ransomware threat. I also want to closely look at a curious WiFi bug that was recently discovered in iOS and what it almost certainly means about the way we're still programming today. Under our Miscellany topic I want to share the SHA-256 hash of the developer release .ISO of Windows 11 that Paul Thurrott, I and many others have been playing with this past week. I have a tip about creating an offline account and restoring Windows 10's traditional Start Menu under Windows 11.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-824.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-824-lq.mp3

A new purpose has also been discovered for this podcast which I want to share, and I've decided to explain in more detail than I have before what I've been doing with SpinRite's evolution it's much more than anyone might expect yet no more than is necessary. Then we're going to conclude with the view of ransomware from Russia, from two Russian security researchers who believe they know exactly why the Avaddon ransomware as a service decided to shutter its operations and publish its keys.

SHOW TEASE: It's time for Security Now!. We've got a great show for you. Steve's got a big SpinRite update. All the deets coming up. There's been another Chrome zero-day. What is that, the sixth or seventh this year alone? And then we're going to talk about ransomware, the economics of ransomware. So two reports, one on how ransomware works and the other on how ransomware figures out how much to charge you. All that's coming up and a whole lot more, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 824, recorded Tuesday, June 22nd, 2021: Avaddon Ransonomics.

It's time for Security Now!. Ladies and gentlemen, the moment you've all been waiting for, the host of our show, the star, Mr. Steve Gibson, here to protect you online. Hello, Steve.

Steve Gibson: Hello, my friend. Great to be with you for our, oh, it's not our last, our penultimate podcast of June, having learned a few years ago what "penultimate" actually means.

Leo: Now he uses it every chance he gets.

Steve: That's right. So this is #824 for the 22nd. I titled this "Avaddon Ransonomics." I had to look at that Ransonomics word a few times. It's what the Russians who wrote the article we will be talking about mostly used. Very interesting. Another less-in-the-spotlight, but still a little too much, major Ransomware as a Service enterprise has shuttered themselves. And what's interesting about this one, we have a lot of - I pulled from a bunch of different sources before I quote some Russians that I found. Another Ransomware as a Service group saying we're going to stop this.

Now, it's also been noted, and I did not have this in my notes, or didn't make it into the notes, that it looks like when these guys shutter themselves, there's some notion of an opportunity vacuum that's created which other groups step in to fill. So it's not like the problem's all gone away. And it also may be that this is just a change of identity. Like they're saying, oh, sorry about that, blocking all of your gas and...

Leo: Dave's not here, man. Dave's not here.

Steve: Yeah, exactly. And then they pop up under another name. So anyway, we're going to talk about that. But first this week, believe it or not, we have yet another zero-day stomped out in Chrome.

Leo: What? My god.

Steve: Yeah, I know. It's tough being number one. We also have some additional intelligence about the evolution of the ransomware threat that I want to share. I want to closely look, and I know that you guys talked about it, although I didn't hear it, but I know you did on MacBreak Weekly because the percent esses were part of the proposed names for...

Leo: Such a hoot.

Steve: Yes, it is. So a curious WiFi bug that was recently discovered in iOS and what it almost certainly means about the way we're still programming today. I'm sure I'll have a different take on it than you guys did, just because I'm a coder.

Under our Miscellany topic, I want to share the SHA-256 hash of the developer release ISO of Windows 11, which Paul Thurrott on Wednesday, I, and many others on the Internet, and many in GRC's newsgroup, have been playing around with this past week. And I'm sharing the SHA-256 hash so that if somebody wants to get it, they'll know that they've got a valid one. I also have a tip about creating an offline account under 11 and also restoring the traditional Start Menu, which those who are using Windows 10 have gotten used to and who are grumbling about the way they changed everything in Windows 11. Anyway, we'll be talking about Win11. So rather than being like the author of Never10, as I famously was before, now I'm ahead of the game.

Leo: Yeah, what the hell? This is not the Steve Gibson I grew up with.

Steve: So a new purpose, Leo, has also been discovered, and this is big news, for this podcast, which I want to share.

Leo: Have you discovered your special purpose?

Steve: I've discovered an unexpected application for the podcast. And I've decided to explain in a little more detail than I have before - because I've received some people saying "Where the hell is SpinRite 6.1?" - what I've been doing with SpinRite's evolution, which it's much more than anyone might expect, yet no more than is necessary. But it turns out what's necessary is more than a point release. But that's what I promised. Anyway, then we're going to conclude with the view from Russia. I almost titled this "From Russia With Love," but I thought, well, I'd already hit print on the PDF.

Leo: Oh, that would have been fun.

Steve: That would have been fun. Anyway, two Russian security researchers who believe they know exactly why the Avaddon Ransomware as a Service decided to shut down, and gave just shy of 3,000 of its attacked keys to BleepingComputer. Lawrence Abrams has them.

Leo: Good.

Steve: And of course we have a fun Picture of the Week that is apropos of what we'll be discussing when we get to Windows 11. So I think another great podcast for our listeners.

Leo: Nice. Lots of good stuff coming up. All right.

Steve: Okay. Our Picture of the Week.

Leo: Picture of the Week, yes.

Steve: I gave this the caption "Windows 11," and everyone will see why. We have a four-frame cartoon. The first frame we have the guy saying, "Let's make these changes to the current system." And he's holding out what looks like a long list of things. And she is now holding the list, and she says, "The current system works just fine." And then in the third frame, "Why do you keep making pointless changes?" And of course the answer is - he sort of is walking away, looking a little downtrodden, saying, "To keep justifying my ongoing employment." So uh-huh.

Leo: Uh-huh.

Steve: Why do you keep making pointless changes? Well, we'll be talking about Windows 11 and its pointless changes.

Leo: I think you're actually closer than - this is actually closer to the truth than people might really understand, yeah.

Steve: Yes. Are we done yet? Oh, no, boss. Oh, no. No, no. No.

Leo: Got to justify my existence. I'm busy. I'm busy.

Steve: Yeah, I've got a whole new look for the trash can, so hold on. Okay. So another day, another Chrome zero-day. As I said last week, this is what it's like to be the world's number one web browser. With glory comes some bruising. We're not yet finished with the first half of the year, yet CVE-2021-30554 is the seventh actively exploited in the wild zero-day that the Chromium team has patched so far this year. We're now at v91, and some other stuff in the middle, and it ends in .114, for all three desktops, which was released last week. It resolves four security vulnerabilities including that -30554 which was a high-severity, use-after-free vulnerability occurring in WebGL, the Web Graphics Library, which is a JavaScript API used for rendering interactive 2D and 3D graphics in the browser.

As it's formally described - and, you know, this is the boilerplate for these things. "The successful exploitation of the flaw could mean corruption of data, which might lead to a crash, and even execution of unauthorized code or commands." Okay. But we know that the bad guys are not going to be interested in corrupting some data or crashing the user's browser. They want executing some commands of their own choosing. Since this vulnerability was being actively exploited in the wild, in this instance we can ignore the lesser possibilities for abuse and go right to remote code execution as having been achieved and exploited.

And if the CVE number 30554 sounds familiar, that may be because 30551 was the previous zero-day, fixed just 10 days earlier. This particular issue was reported to Google anonymously exactly one week ago, on the 15th. And this .114 release of Chrome occurred just two days later, on Thursday the 17th. This highlights the point I made last week about the turnaround speed that's required from today's web browser deployment developers. They don't sleep so that we can.

In the show notes I have the six previous zero-days. There was one on February 4th, then the next one on March 2nd, then March 12th. Then we have April 13th, April 20th. May somehow skated by because we had one in late April and one in early June. And then the last one, the sixth one, on June 9th.

So Shane Huntley, the director of Google's Threat Analysis Group, tweeted two weeks ago on the 8th, so that was the day before the sixth zero-day. And he said something at the end I don't quite understand. Maybe you can figure out what he means, Leo. His tweet was: "I'm happy we're getting better at detecting these exploits and the great partnerships we have to get the vulnerabilities patched. But I remain concerned about how many are being discovered on an ongoing basis and the role of commercial providers."

Leo: You mean like Google? Who do you mean?

Steve: Yeah, I don't know what that - and "the role of commercial providers."

Leo: Maybe he's not talking about Chrome exploits. I mean, that's bizarre.

Steve: Isn't that weird? It's like...

Leo: Commercial providers.

Steve: And so I put a link to his...

Leo: Oh, I know what it is. Maybe not all of these are being discovered by Google's Project Zero, but by other security providers, like Kaspersky and so forth. And that's who he's thanking.

Steve: Ah. "But I remain concerned about how many are being discovered on an ongoing basis and the role of commercial providers."

Leo: Well, he shouldn't be concerned about that. He should be grateful.

Steve: Yeah, yeah.

Leo: Maybe he means commercial malware providers. I don't know what he's talking about. That makes no sense at all.

Steve: Yeah. So anyway, I put his Twitter feed link in the show notes. And I thought, what? And so I went there and like read around his other, like maybe I could get a clue from what he'd said before or after that tweet. So I found it on June 8th and read in the region, but no. I have no idea what he meant. So he is a character. His photo made me think, like, well, okay, you do look like...

Leo: That's a good-looking fella. Yeah.

Steve: Anyway, so thank you, Shane, for these last five minutes of the podcast. We have no idea what it is you're talking about.

Leo: Go back to work, Shane.

Steve: But for what it's worth, we're concerned, too, about how many are being discovered on an ongoing basis because you're breaking records, and not the kind we'd like you to be breaking. Keep finding them. But how about stop making them? That'd be better.

Okay. So as we have been noting, ransomware perpetrators are increasingly purchasing their access. The security firm Proofpoint has been tracking the ransomware underground for many years. And I meant to put a link to their posting because they had a cool graphic, like showing all the interconnectivity of all the actors, TA-850, you know, Threat Actor 850 and Threat Actor blah blah blah. And then, like, which malware droppers they were using and who they were attacking and blah blah blah. Anyway, last Wednesday they released a report titled "The First Step: Initial Access Leads to Ransomware."

The report detailed the means by which ransomware attackers are increasingly partnering with unaffiliated cybercrime groups to obtain access to high-profile targets. Oh, you found it, yay. Perfect. This is the trend that we've talked about previously. This was the way we believe the Colonial Pipeline attack began; remember? An existing VPN logon credential was purchased on the dark web by a DarkSide affiliate, and that was used to gain entry into Colonial Pipeline's internal network. Proofpoint's research confirms this trend, but puts a little more meat on the bone.

They explain that, they said: "Ransomware threat actors currently carry out" - and oh, by the way, this is a new term, "big game hunting," so we'll be encountering that term a little bit later - "carry out big game hunting," although I have a feeling that that's the trend that's going to be changing as a consequence of the fact that the big game turned out to be loaded for bear and could shoot back. Anyway, "...conducting open-source surveillance to identify high-value organizations, susceptible targets, and companies' likely willingness to pay a ransom." I thought that was interesting.

Leo: This animated GIF is cracking me up. The hacker apparently does this over several days period of time. I'm sorry, I didn't mean to interrupt, but I'm watching this. It's bizarre, the way they're attacking. Go ahead, I'm sorry.

Steve: So they said: "Ransomware threat actors can leverage existing malware backdoors to" - oh, I'm sorry. "Working with initial access brokers" - that's the other new term. So we have big game hunters and initial access brokers. "Working with initial access brokers, ransomware threat actors can leverage existing malware backdoors to enable lateral movement and full domain compromise before successful encryption." They said: "An attack chain leveraging initial access brokers could look like the following." And so they have seven points.

First, a threat actor sends emails containing a malicious Office document. Number two, a user downloads the document and enables macros, which drops a malware payload. This is like exactly the scenario we've highlighted in the last couple weeks. Three, the actor leverages the backdoor access to exfiltrate system information. And four, at this point the initial access broker can sell that access to another threat actor.

Leo: Ah. So he's already in, but he's not going to do anything with it himself.

Steve: Right. Exactly.

Leo: Wow.

Steve: So five, the actor then who purchased the access from the initial access broker deploys Cobalt Strike via the malware backdoor access, which enables lateral movement within the network. Six, the actor, meaning the ransomware affiliate, obtains full domain compromise via Active Directory. And then, seven, the actor, the ransomware affiliate, deploys ransomware to all domain-joined workstations.

So just like an organic virus which mutates to improve its chances of survival, we see here a similar mechanism of action. Anything which works to maximize the ill-gotten revenue of malign actors will be reinforced. So in this instance we're seeing growing evidence of increasing specialization within the ransomware business model. We first saw ransomware gangs doing their own work. Then the affiliate model appeared to create much larger and broader ransomware franchises. We expect to soon see more formalized dark web escrow services as uninterested third parties are created to manage and apportion ransom payments among these different actors. And now we're seeing the emergence of IABs (Initial Access Brokers) as the ransomware affiliate role divides and further specializes into initial entry and post-entry exploitation.

For many years on this podcast we've observed the situation that malware was present. Like, a lot of it, everywhere. It would get into a router border device and would typically set up a bot in a router that would contact a command-and-control server to await instructions. And remember how I've said several times something like, "At the moment, the bad guys are focused upon the outside. They appear to be curiously uninterested in whatever network they've gained access to." And I observed that at some point that would change, and that things would then get a lot worse. We're seeing the beginning of that, as those initial access brokers, I mean, even like labeling themselves that, start to inventory the systems they have long had access to, but haven't had any means of monetizing, other than perhaps having the device participate in a cryptocurrency mining pool.

But now the networks behind those routers belonging to corporations of significant but lesser size will be examined as potential plunder targets. One of the lessons that has been learned by the ransomware denizens is that, if you want to remain viable, it's far better to avoid what we might call "the Colonial Pipeline mistake." And we'll be talking about that in a second, toward the end of the podcast.

So basically what that means is that attempting to hold infrastructure at ransom, while it may appear at first to be the mother lode, brings with it far too much unwanted political and law enforcement attention. It is far better to sneak around under the radar, siphoning off and aggregating many more much smaller ransoms. The REvil gang's subsequent attack on JBS Meat Packing was another such mistake. Sure, they netted $11 million dollars, but they also got the U.S. to start considering ransomware to be terrorism. And while Putin may bluster, shrug, and attempt to laugh it off, you have to know that he would have been made uncomfortable by the U.S. President facing him down one-on-one and making clear that this will not be allowed to continue.

My point is that carrying out 11 $1 million attacks against non-name-brand targets who no one has ever heard of would have been far wiser in the long run. The ransomware affiliate model, enhanced with initial access brokers and third-party escrows, is evolving to enable exactly this. It allows for scaling up the number of attacks while maintaining efficiency as the size of individual attacks is reduced for the purpose of staying clear. This creates a blur which neither politicians nor law enforcement will lock onto the way they locked onto Colonial Pipeline and JBS. It will tend to make many fewer headlines, and that's the point.

And remember back a few years ago when we were talking about how the networks of managed service providers were being compromised; and their clients, and one example stands out, like networks of dental offices were being held for ransom. Those were much less sexy attacks, and no one cared much. The local FBI would have been engaged, but those attackers never became big news. We watched them on this podcast and were aware of them, but they were not cocktail party and casual dinner conversation.

I said earlier that anything which works to maximize the ill-gotten revenue of malign actors will be reinforced. The clear takeaway from the recent high-profile attacks is that those were a mistake, and we have to know that the entire ransomware industry watched and learned. What they learned was that the way to get rich is to streamline the system. Don't attack big. Attack small and attack more. Distribute the pain and distribute the influx of cryptocurrency. To remain under the radar is to remain in business. The gangs that learn that lesson and keep their money-grubbing affiliates in check are the ones who will remain active in the long run.

Okay. So a weird and fun bug hit iOS. An interesting and somewhat humorous bug was discovered in iOS's parsing of network SSIDs. There's long been a concept in programming languages, at least since FORTRAN because I recall it being there, of using a so-called format string to describe the shape of the contents of either incoming input to a computer, or the way some output variables should be formatted for presentation on output. So, for example, an output greeting might be formatted as a string, "Hello %s," where when that format string is used, the "%s" tells the computer that the next argument to the function is assumed to be a pointer to a string.

In this case, the "%" is known as an escape character, and the character or characters that immediately follow it specify the details of the format. The "%" is called an "escape" because it signals the text parser to stop treating the input string at that point as literal text sent to the output, and instead, to insert some special formatting control. So, for example, "%d" might tell the parser to treat another argument as a date and to format it accordingly. And if the programmer wants to actually output a "%," then "%%" is often used.

Okay. With that bit of background, a security researcher who was poking around at iOS somehow discovered, and don't ask me how, he's like, well, let's try this, see what happens. He discovered that if a WiFi network's SSID name, you know, the name that you see in public when it comes up in a list of you can join any of these networks. So that's the name that the beacon is broadcasting, the SSID. If it is set to "%p%s%s%s%s%n" - so percent sign p, then four esses, percent sign esses, and then percent sign n - and an iOS device then attempts to join any WiFi network having that name, the device's WiFi would become immediately and semi-permanently inoperative. A restart/reboot would have no effect, and all logon attempts to reverse the change would fail. Any attempt to reenable the WiFi subsystem to fix the trouble would immediately crash before the user could use the subsystem to resolve the problem.

So this is one of those things that's sort of our collective fault in the security business, and more broadly the programming business, really, by choosing a programming design pattern which places convenience way in front of security. One of the things that must be done, when a string that's under the user's control might be processed by code that's parsing for escape sequences, is for the user-provided string to be explicitly "de-escaped" first. In the example case I provided above, this would mean doubling up any "%" characters so that they would be seen as the percents that they were apparently intended to be, rather than as the active "escape sequence" which would cause the parser to reach for a subsequently provided argument which, in this case, would be absent and would almost certainly lead to a crash. So what must first have happened is that the SSID string which itself looked like escape-formatted text, confused some formatting parser somewhere in iOS and likely caused the internal WiFi system to completely hard crash, and crash and crash and crash.

So the concern was that, as news of this spread - and, I mean, every tech site that I saw had fun spreading the news. The concern was that, as news of this spread, annoying jerks would immediately begin exploiting the discovery. And indeed that did happen. Postings began to appear telling naive users that they could obtain much faster WiFi by renaming their access points accordingly. Shhh. Don't tell anyone. And of course when they did that, all their iOS devices crashed. Fortunately, after some additional experimenting, it was discovered that the devices' WiFi function could be restored by going to Settings > General > Reset > Reset Network Settings. That would flush out all of the existing sticky stuff and resolve the problem.

So how did we get into this trouble in the first place? We would really have to say that it was lazy language design. The practice of mixing special-meaning control text in with literal text is nothing less than a kludge. Some form of separate out-of-band specification should be used to govern such formatting. Mixing those functions together into a single stream is a recipe for disaster. So why do we do it? We do it because it is so much easier to do it that way. And as a result, many languages do.

Recall that when an HTTP GET query contains arguments, they take the form of a question mark followed by the name, an equal sign, and then a value, so-called "name=value pairs," and they're joined with an equal sign. And those name-value pairs are separated by ampersands. But what do you do, how do you have a name or a value containing an equal sign or an ampersand? Once again, we're mixing literal text with control characters. It can be done safely. I lived in that world throughout SQRL's development since there was a lot of that going on. And I was terrified of making any mistakes there because they would almost certainly be devastating and because, you know, it's so easy to make a mistake. Languages, protocols should not be designed so that it is easy to, I mean, like you almost have to do it wrong. And it's so difficult to do it right.

And indeed, countless mistakes have been made through the years with this HTTP formatting since the very beginning by web programmers who were not being sufficiently concerned and cautious. Because, you know, the system is begging you to make a mistake. And while I'm certain that the details of iOS's WiFi are different in detail, and that the problem will be trivial to repair, it does appear that exactly this problem is what just bit Apple.

Leo: So is - pardon me? Go ahead.

Steve: I was going to ask if the MacBreak Weekly guys had any additional specific information.

Leo: No, I explained that it was a format string. %p is a pointer. So what you would do normally is a printf, and you'd put the format string in, and then you'd follow it by the things that correspond to each of the items.

Steve: Of the arguments; right.

Leo: You'd have the pointer. %s is an alternative string, so you'd have four strings. And then you'd have a new line. I'm guessing that what that actually got interpreted is as a pointer with four zeroes, in other words, a null pointer; right? And that crashed it. But who knows? Maybe they do some URL encoding with it? They're doing something weird with it. And you're right, why aren't they sanitizing those inputs? It's easy.

Steve: Yeah, yeah. I mean...

Leo: You don't trust everything that comes in over the transom.

Steve: Yeah. Again, and that's the point, is you shouldn't have to, the language design should not be such that it's this easy to make that mistake. This mistake keeps happening over and over and over because the language design is insecure. It's lazy. It's so easy and convenient that that's what many languages do.

Leo: And I'm guessing that it is C that whatever code is handling it is written in, or Objective C maybe. But that was the whole point of C was it lets you do anything. Dereference pointers, reference pointers, point to any part of memory, you know, allocate memory at random, overrun it at any point. That's the point of C. C lets you do anything. It's very powerful.

Steve: Right. And it's very popular because programmers are jocks.

Leo: We love it. C's great.

Steve: You know, get out of my way. I don't want any garbage...

Leo: And I won't make a mistake.

Steve: I want to collect my own garbage.

Leo: Yeah. Stay out of my garbage.

Steve: So anyway, again, this is like, for our podcast, such a perfect example of something that should not have happened because it shouldn't be up to the programmer to be careful against the built-in mechanisms of the language they're using. They shouldn't be using a language that can do this that way. But we are.

So a little bit of miscellany. Paul Thurrott, I, and many of those in GRC's newsgroups, and most of the rest of the world who have been curious, have been playing around with Windows 11 since last Tuesday when a development release ISO image first appeared on the Internet. The moment I saw that the ISO had leaked I went searching for and found it. Links were not difficult to find. And I'm sure that they're still not today, as this has spread now even further and wider than it had a week ago. A couple of thoughtful listeners also DM'd links they had found.

For the most part, the links tend to be transient since they're being removed by the powers that be as soon as they're found. But even if I had stable links, I would be uncomfortable using this podcast to reshare them since Windows 11 isn't officially released, and its sharing could reasonably be considered promoting the use of what is essentially pirated commercial software. The fact that it's been, and is being, so widely pirated doesn't make it any less so.

What I can do, however, is to protect our listeners. We can offer our inquisitive listeners some protection in the form of the SHA-256 hash of the known-to-be-valid ISO file. That way, anyone here who believes they may have obtained the ISO from who knows where can readily verify its validity and safety for themselves. Being very cautious myself, last week I downloaded many of these 4.8GB apparent Win11 .ISOs from very different sources and checked each one's SHA-256 hash. They all matched. So they were all valid. And I also confirmed that with some other hashes that others were also making.

This SHA-256 hash in the show notes, I've got the whole thing there. Its first 32 bits, it starts out with b8426650, and its last 32 bits, it ends with 6da60dcb. Those 64 bits alone, the first 32 and the last 32, provide 1 in 18.4 times 10^18 verification strength. So if your hash begins and ends with those, you're okay.

For anyone who's interested in making a hash, there are various Explorer shell context extensions to show the hashes of files which are right-clicked on. But Windows has two built-in commands to do the same job. There's a standard command prompt command. It turns out that the Windows certutil (C-E-R-T-U-T-I-L) has a hashfile forming function. You say certutil -hashfile, and then the filename, and then SHA-256. I've got this shown in the show notes also. And then PowerShell has a built-in function, and it defaults to SHA-256. It's Get-FileHash, then the filename. Hit Enter. It takes a while for a 4.8GB file. But in both cases you'll get the hash.

And I haven't seen any instances of malicious ISOs. But you've got to know somebody is going to create one. So if you do go get this, and if you're curious, take the time to make sure that the hash matches the one that I've listed. And having played around with it, it is indeed Windows 11. And I have to say, oh, it is truly gorgeous. It is very pretty. But Leo, I'm a bit disappointed because I keep encountering plenty of pointy corners.

Leo: It's supposed to be rounded.

Steve: Exactly. Now, it occurred to me that it might be that the extremely high resolution of my monitor is not being compensated for, so that the subtle visual rounding is being lost. I believe that on that machine I have my text-scaling set at least to 150 because the resolution is so high. So what that would mean is, if they're rounding with a bitmap, then they're not scaling the rounding as they're scaling the text in order to keep the effect of the rounding the same. Maybe they'll fix that. But for whatever reason, what I'm seeing still seems quite pointy in places. There are reports of Win11 install failures on older machines. And when I heard you and Paul and Mary Jo talking about it last Wednesday, Leo, one of the reasons they were conjecturing that Microsoft may have done a Windows 11 is to promote the sale of new hardware.

Leo: Yeah.

Steve: It's like, hey, you want 11? Get it on this new machine. But what's curious about that is it's certainly going to be able to upgrade over 10. So one of the things that's interesting is that Windows 11 refuses to run due to the desktop or whatever machine it's running on, but probably a desktop because laptops are better this way, not having a Trusted Platform Module, not having TPM, and thus being unable to support Windows Secure Boot, which requires a TPM. Support for Secure Boot has always been optional for Windows 10. But we have heard that, and now the industry is confirming, that for some reason Microsoft has decided to make it mandatory, that is, Windows 11 must have a Trusted Platform Module. So that would rule out its use, if it hadn't already been worked around. And maybe they'll stomp that out by the time it gets to release.

Leo: Remember, you're not using a release version. In fact, there's a lot of things missing from what you're using. We know that.

Steve: It is a dev release. So anyone running Windows can first of all check for the presence of their hardware platform's TPM by running a little snap-in. It's just called tpm.msc. So hit the Windows+R, you know, the Windows key and then R, to open the Run dialog. Then enter tpm.msc and hit Enter. That'll launch the TPM configuration applet that's built into Windows - 7 has it, and I'm sure everything since does - and see whether the system agrees that it's got a TPM. That'll tell you whether that platform will eventually run Windows 11.

I have links to two instances of web pages talking about the problems people have had last week installing Windows 11. Fossbytes.com is one of them. "Solve TPM 2.0 error installing Windows 11 fixed." And basically it involves taking one particular file from the Windows 10 and substituting for Windows 11. Maybe Windows 11 will end up fixing that, double-checking for it. Depends upon how much Microsoft cares about locking this down. I don't know what their position will be.

Oh, and one other really interesting thing. I'm sharing this mostly because it came from one of our own very active people in GRC's newsgroup, where he discovered this. The other potential gotcha hit people who do not wish to log into Microsoft during the installation and - I didn't want to because, I mean, this wasn't official, right, Windows 11. I didn't want to be like, oh, yeah, I'm going to log on with my Microsoft account. No. But so you'd like to create an offline account.

Although Windows 10 Pro allows for bypassing this in the setup UI, the Home edition of Windows 11 does not. So if Windows 11 is being installed on actual hardware containing a built-in Win10 Home OEM key, it won't give you the option of which version of Windows 11 you want to install, as I got when I installed it in a virtual box VM. Instead, it'll insist upon installing the Home edition, and that in turn insists upon the creation of an online account. In Win11 Pro, it offers the "I don't have Internet" option to bypass the establishment of an Internet connection. But that's been deliberately removed from the options for the Home edition.

Okay, so here's the surprise. There's a workaround. A guy named Adam, whose moniker is Warwagon in the grc.securitynow newsgroup, discovered that it's possible to simply close the insistent "Let's connect you to the Internet" dialog by hitting the ALT+F4 key combination. Well, anyone who's used Windows and likes the keyboard knows that ALT+F4 is the quick shortcut for killing an app. It's the same as going up into the context menu and closing, you know, Close or Exit or whatever. It just terminates the app. So Adam discovered that that works for this "Let's connect you to the Internet" dialog.

Oh, he also told the guys at Neowin, and they wrote the following. They said: "However, here's where Adam's simple workaround came in handy, which is both amusing and surprising. When Windows 11 Home prompts users to connect to a network, a simple ALT+F4 shortcut closes the prompt, and the screen proceeds directly to the local account creation page, something that is never offered to users in the usual process. This bypasses the entire Microsoft account login screen, which is a nifty little trick for those who want to avoid signing into their accounts during the Out of Box Experience (OOBE) process, especially in these early days when most installs of the OS are happening on virtual machines." So actually under a virtual machine you do get the choice of which version you want to install. I chose Pro, but you can choose anything you want. So anyway, just a neat little tip.

Oh, and those who are not fans of the new positioning and look of the Windows 11 Start Menu, you may know that it creates - now there's like the bar of items that you can click on that used to be over on the left, the shortcuts, they now float in the middle, continually recentering themselves as new additions editions appear. And the Start Menu is on the left but floating with them in this group, looking very much like a Macintosh Dock. There is a reg key setting, HKEY_CURRENT_USER

\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced. And once you get to the Advanced key, there's a whole bunch of little settings. You will see Start_ShowClassicMode, which will currently have a zero in it. Just change it to a one, reboot, and then you get, as the name sounds, ShowClassicMode; you get back to the menu you're used to from Windows 10.

And finally, Leo, before we take our second break, we have the new purpose that has been identified for the Security Now! podcast.

Leo: Oh, yeah? What's that?

Steve: This was tweeted from someone whose moniker is Lost World, @Bruin144. He wrote: "Security Now!. Just the sound of the podcast by itself removes intruders."

Leo: What?

Steve: "For the second time," he wrote, "in a few years I have used Security Now! playing on an endless loop to remove intruders - raccoons - from my attic. A few years ago a mother raccoon and her babies invaded an inaccessible-to-humans part of my attic. I put a speaker where she could hear it, and because raccoons don't like the sound of people talking, you and Leo prompted her to decamp."

Leo: Why did he include me in this? It could have just been you.

Steve: "This week a new raccoon defeated my security measures and moved in. Eighteen hours of Security Now! later..."

Leo: Chase any varmint away.

Steve: "...he moved on and out, and I repaired the mesh he had pulled down."

Leo: That's hysterical. Wow.

Steve: So just a tip to our listeners. If you have a problem with raccoons...

Leo: Security Now!. It rids you of raccoons.

Steve: In less than a day you will be raccoon-free.

Leo: That's hysterical. Hey, I want to correct myself. We were talking about that SSID string. And in C++ and C and Python and many languages...

Steve: "\n."

Leo: "\n." A "%n" is worse. The last character in that string, "%n," the corresponding argument - I'm reading from the C++ documentation. The corresponding argument must be a pointer to assigned int. The number of characters written so far is stored to that location.

Steve: Ooh, god.

Leo: So I'm guessing...

Steve: Who designed C++? Oh.

Leo: I mean, you should not be able to printf to a memory location. But that's what this does. So I'm guessing that really is - that last character there, that's the real killer.

Steve: It's surprising you still have a phone after that happened.

Leo: So "%p" is a pointer, "%s" is a string, "%n" doesn't print anything at all, it just stores what's come before it into an arbitrary memory location.

Steve: Oh, oh, god, just shoot me now.

Leo: No wonder it crashed the phone. Good lord. And you're right. Why would printf have that capability? It turns printf into a poke, basically. Crazy. Crazy.

Steve: So wsprintf would send that to a buffer.

Leo: Right.

Steve: So you might want to know how many characters had been stored in the buffer in order for like centering text on a line, that sort of thing.

Leo: Oh, yeah. But I bet you people use it for a poke.

Steve: But again, talk about dangerous.

Leo: Yeah.

Steve: Talk about dangerous, it's like, yeah, no.

Leo: No. I think I would always assume that these printf commands print to standard out, not to memory. That's terrifying. Ws does.

Steve: Wsprintf does exactly that, yeah.

Leo: Interesting. I never use that one.

Steve: And it uses the same format string.

Leo: Yeah. Well, that's probably it is that these format strings are general. They're generalized for all different kinds of stuff. Wow. Just crazy. Crazy.

Steve: So that people realize I was being rhetorical...

Leo: Oh, you've got it there. You're not making that up. He's got it.

Steve: No, this is Stroustrup's...

Leo: It's Stroustrup's book.

Steve: ...original, yes, C++ book.

Leo: Wow. Show the binding. That's too thin. That can't be the real thing. Where's the - I have the other one that's really thick.

Steve: Well, this one was the original 1991 book.

Leo: Oh, okay.

Steve: So, you know...

Leo: My C++ book from Stroustrup is like a doorstop, which is all it's good for.

Steve: This is the spec.

Leo: Oh, I get it. So it is, it's very terse, sure.

Steve: Yeah, yeah, yeah. I mean, it is, as it says down there, ANSI-based document.

Leo: Oh, that's why. Yeah, my book is actually, like, here's how you learn C++. Which I quickly gave up on, by the way. What a nasty language.

Steve: You should try to forget it, if you can.

Leo: I loved C. Loved it.

Steve: I do. And C++ is largely now regarded as a mistake. So it's like, well, okay.

Leo: I bet it's still the number one language, though. Maybe JavaScript or Java have superseded. I don't know.

Steve: Yeah, actually I think JavaScript is, last I saw. Because we've talked about that from time to time on the podcast.

Okay. So I received a Twitter DM last week where one of our listeners asked, "What the hell is going on with SpinRite, and when are we going to get it?" So, you know, after my working on SQRL, apparently without end, I think that's a reasonable question. And it reminded me that everyone in GRC's spinrite.dev newsgroup knows exactly what's going on and where things stand because I periodically let them know. But that's a small fraction of the people we have listening to the podcast.

And since everyone owns local mass storage that they care about, and since I know that many of our listeners own SpinRite 6 and are looking forward to this next release, I suspect that it would be reasonable to assume that most of our listeners would be interested in having some better sense for what is happening while they are being patient. This guy not so much. And because this v6.1 project has grown into so much more than I expected, I wanted to take a bit of time today to provide a bit more visibility into what I've been up to.

The last incremental development release of the work on SpinRite had the new SpinRite code finding all of the system's mass storage devices, regardless of how they were interconnected to the system. It also determined the most comprehensive way that each device could be interfaced. For example, a SATA drive attached to an AHCI controller port will be visible through the BIOS, and perhaps with BIOS extensions. But it will also be visible directly through its hardware, which SpinRite is now able to access directly.

So the last test that I released for everyone to play with was enumerating all the drives and showing all of each drive's relevant data far more comprehensively than any previous release of SpinRite had. That release went amazingly well, actually much better than I expected, probably because we've been moving forward carefully and leave only fully tested and verified code in our wake.

After that, the next thing I had planned to do was to bring the drive benchmarking system online, since SpinRite's built-in read performance benchmarking would have to be testing the read-channel of every drive. So exactly one month ago today, on May 22nd, I posted an update under the subject "One thing leads to another," which was the beginning of the trouble I found myself stepping into. So I want to share that post and the four subsequent posts I've made since about what I'm doing with SpinRite. Even if you're not interested in getting your hands on v6.1, anyone who's interested in computer technology will probably find this little bit of snapshot interesting.

So May 22nd, "One Thing Leads to Another." I wrote: "Gang. It occurred to me last evening that by the time I have the benchmarking system running for everyone to test, this thing will be close to finished. Everything is interconnected, and one thing led to another. I said earlier that in order for the benchmark to be able to read sectors from the drive, it would need to have a lot of the system rewritten and running. That's turning out to be more true than I realized at the time. One thing leads to another, and I've been busily following those leads.

"A very useful thing has been that I know what lies beyond v6.1. That's coloring many of the decisions I'm making along the way. I'm not really writing just for this, but also to a very large degree for this to be a near-term future where we're joined by native drivers for USB and NVMe. This next SpinRite will have an architecture that's ready to accept them.

"Yesterday, I came up with a slick encapsulation for SpinRite's IO work: a single function that replaces all of the scattered IO throughout SpinRite. The problem was that sometimes SpinRite needs to use Real Mode 16-bit segment:offset buffers in low memory." For example, the BIOS only knows how to transfer there, and sometimes it's able to use a 32-bit linear buffer in high memory, for example, when I'm able to use the hardware directly. "And in order to reduce the consumption of low memory, some working buffers that used to be in low memory can now be moved into high memory since, for the first time ever, SpinRite 6.1 will be running in flat real mode."

So I explain: "So each drive seen by SpinRite can be one of five different access classes: BIOS only, Extended BIOS without hardware access details, Extended BIOS with direct hardware access details, an IDE or SATA drive on the PCI bus with Bus Mastering, or a SATA drive on PCI with an AHCI controller." And then each of those, I've got little notes after each of those five which shows which of those the following six things apply to: Limited to max CHS, that's Cylinder Head Sector size, which is 28 bits, 137GB; potentially any size drive; access to the entire drive plus SMART and the SMART Log data; segmented memory transfers only, that is, lower 16-bit region, below 1MB; possible use of large sector count transfers and linear RAM in high memory; and 16MB (32k sector) transfers to high memory. So you actually sort of end up with a grid of different access types and then the details that each of those access types applies to. This all needs to get figured out.

So I said I already had the concept of "selecting a drive into context" which has traditionally been shown by SpinRite's "Selecting Drive for Use" screen. That concept has matured significantly to become much more generalized, flexible, and comprehensive. Yesterday I realized that all of SpinRite's data transfer work could be merged into just three generic types: Transfer to/from a single-sector buffer, transfer to/from a single-sector scratch buffer, and transfer to/from a track buffer. In each instance, the location of the buffer, the means of performing the transfer and of obtaining the results will depend upon the drive that's currently selected into context, the details of that specific drive, and the hosting system's BIOS features.

So this new function, named just "IO," will encapsulate all of those specifics. I call it with a function code to specify the type of transfer I want. It obtains the starting sector number, which is now 64 bits, from global memory. And the length of the transfer is implied by the operation, a single sector or an entire track; but then the length of the entire track is also obtained from memory, and by the specific drive's characteristics. There will also be a generic MOVE function to move data between the sector buffer and the track buffer, and my code will not need to worry about which of the buffers are in use because the context will know.

So then I explained SpinRite never had, nor needed, anything like this before since until v6, everything ran through the BIOS exclusively, and at v6 the BIOS was only being bypassed for a subset of special case accesses, that is, all the extra data recovery that I was able to add to v6. And I finished with: "The beauty of this abstraction is that it cleanly divides the drive characterization, which populates the drive features database, and all subsequent drive access IO from the logical operational parts of SpinRite. And moving forward into the future, when things like subtle read-timing features are added, the functions offered by that single IO function can be augmented."

So that was the end of the first post. And essentially what I've done is I've used an object-oriented philosophy to create a clean demarcation between the way the IO is being done to individual drives by the way they're connected and the technology, the best technology that I have to access that drive. That's one side of the divider. The other is all of the stuff SpinRite needs to do to access the drive. But those things don't change depending upon the drive. So I'm able to create an abstract function, which is what I have, this IO function, that allows that.

Then on the 29th of May - that was on the 22nd. So a week later, on the 29th, one of the things that happened as I was working through the code, I was looking at moving one of the big buffers that's in low memory into high memory because that just frees up low memory, which is still resource constrained. I ran across a whole bunch of code for the logging system. And I looked at it, and I was on the mission of, like, changing that from running on a low memory-located log to high memory. But I stopped myself and said, wait a minute. Do I even want this anymore? Because the way it has always worked is that SpinRite would maintain a log in the root of the drive that it was working on, and the user could specify how many previous logs to retain.

So all this code that I was looking at, and there was a bunch of it, it looked at the old log, and it parsed the old log to count, like to find where was the beginning of the first log entry that was older than the user's configuration setting now said they wanted so that it would only keep the most recent N log entries. And then I would take that, and I would get rid of the older things, rewrite the log file, moving the most recent N logs down and then be appending the new log freshly to the top of the log. And I thought, really? So I put that out to the group, the grc.spinrite.dev group. I don't think I've ever had 100% consensus on, no, this is not what we want anymore.

So the new SpinRite 6.1's new logging approach just does sequentially numbered logs. It looks at the highest numbered log file it can find on the boot media, which is where it logs back to, and just chooses the next one. You know, it's time and date stamped because it's a file, one file per drive. You'll have the option of making it one file per use of SpinRite, since one use of SpinRite could run on multiple drives, that's the only choice you have, instead of retain N prior logs, which didn't make any sense to anybody anymore. So anyway, that was one of the entries. And in the process I simplified the code. I just threw away all that old code that was doing all this crazy log file manipulation, and I don't have to bother with it now.

On June 9th I said: "Gang." Oh, this was "SR v6.1 Progress Report," June 9th, so earlier this month, a little over two weeks ago. I wrote: "I'm glad I'm doing this. It's tedious, but necessary. Everywhere I turn, the code needs to be rewritten or edited. So I'm just plowing forward, fixing everything I encounter. I'm no longer trying to get something for everyone to test. There was too much interdependence for that to be feasible. Or as I wrote before, 'one thing leads to another.'

"In order to catch every instance of something that needs changing, I change the name of a variable to force assembly errors due to the old name no longer existing. That's necessary because the old variable might have been 16 bits, and the new one needs to be 64. But it means that a daunting list of errors results from every reference to the now gone obsolete variable. So I then move through one by one, addressing each of the references to the old variable.

"I'm currently working to fix all references to what was previously 'OperatingLocationLow' and 'OperatingLocationHigh' variables. They were each originally 16 bits back in the pre-32-bit days." Right? Because this thing ran on an 8086 that didn't have any 32-bit math or registers. So I had to have two variables, OperatingLocationLow, OperatingLocationHigh, each 16 bits. And so they are both being replaced by a single 64-bit transfer location variable.

And I said: "I've been working on this one for a few days, and I've whittled the list of assembly errors down to about 25% of what it was initially. When I finally emerge on the other side of the variable updating, we'll have a new foundation for probing generic mass storage. But at the moment I cannot even estimate what percentage of the way through I am. At some point I'll receive a welcome surprise that there are no more errors because all references to the new variables will have been encountered and recoded.

"The problem, of course, is that even though I have and will be as careful as possible, I will have inevitably introduced some new errors. So it will be necessary for me to step through the code to watch everything work at least once. And that's fine too, since once that's done that new mass storage foundation will be real, solid, and functional. And then I'll be able to get back to the area I was excited about earlier, where a single IO abstraction procedure hides all of the details of drive access. Behind it will lie handlers for BIOS, PCI/IDE, and AHCI/SATA drives. And adding handlers for USB and NVMe, and who knows what else in the future, will then be very straightforward."

Two days later, on June 9th, I posted "SpinRite's Source Code Line Counts." I said: "Gang. During my after-dinner walk with Lorrie, I mentioned that once things had stabilized and settled down with SpinRite, that is to say, once I kind of thought I was done, I planned to read the source from top to bottom to find anything that I hadn't addressed. She asked how long the source code was, and I didn't know. So I just did a quick line count to see."

Leo: I wonder what she thought you would answer, like, oh, it's 500 pages? I mean...

Steve: Yeah, she's just like, well, how long is that?

Leo: How long is that?

Steve: Because, you know, someone she cares about is going to read something. How long is that going to take?

Leo: It's surprisingly short. It's really more a short story than a novel, I would say.

Steve: That's right. Well, so in the show notes, for anyone who's interested, I have a link that I posted as part of the posting since the newsgroups are text only. SpinRite currently has a line count of 27,556 lines.

Leo: That's a good number of lines.

Steve: Now, that's not all code, since especially my newer code tends to have longer comment blocks at the top of procedures to explain anything that the procedure's long name doesn't already make clear. At the same time, I heavily comment the ends of my lines, but those won't show up as additional line counts. So it does give some sense for SpinRite's source code base. I excluded a handful of files of UI content - the screens, the screen composition definitions, and the text that fills them, since they aren't code. And most of them aren't changing. Basically, the UI is pretty much the only thing that is surviving this conversion of SpinRite 6 to 6.1. And even that has had a lot of facelift already.

I wrote: "In the beginning, SpinRite was mostly a single SR.ASM file. And you can see that heritage since SR.ASM remains by far the largest single file at 8,652 lines."

Leo: Yeah, I noted that. Is that the main code, kind of?

Steve: Yeah. There used to be like one code file, SR.ASM, and then a whole bunch of UI files. But I've been breaking it apart into smaller, more manageable and functional pieces during this recent work. So things like MATH.ASM and MEM.ASM, which were once part of SR.ASM, are now separate files. I broke them out since they needed lots of reworking and rewriting. You could imagine that all the math had to be updated, and all the memory management needed to be fixed.

Leo: I'd love to see some of this source. You don't ever publish any of it, do you. I know it's proprietary.

Steve: I haven't. Yeah, I haven't. But I wouldn't mind sharing some.

Leo: Just a little bit. Just a little bit. Some of it.

Steve: It is fun to look at it.

Leo: In your will, when you pass on, you know, let's open source it.

Steve: Oh, I'm definitely going to release the source once there's no more ongoing commercial work for me.

Leo: That would be great.

Steve: I've stated that publicly to the gang, so, yeah.

Leo: Excellent. Nice.

Steve: And I'd like it to outlive me. It looks like it might, actually.

Leo: Yeah, well, you've certainly modernized it. Boy.

Steve: Oh, yeah. I'll be proud actually to release it. So, okay. So the final one, my most recent note from last Thursday, June 17th, I said: "Gang." And this was titled "Another Update." I wrote: "I figured that since I've done something else again, I ought to loop everyone in on what's been going on." And I should explain, I mean, I work on this thing 12 hours a day, 14 hours a day. I mean, so for me, even though that was a week ago, the previous update, everything has changed since then. I mean, because so many hours have happened.

So I said: "Gang. I figured that since I've done something else again, I ought to loop everyone in on what's been going on. As I wrote previously, I updated everything to handle the shift from 32-bit sector addressing to 64-bits. After finishing that, I returned to the work on the IO abstractor, which will be providing a uniform interface between SpinRite and all current and future drives and technologies. The trouble I then encountered was what exactly I wanted the abstracted IO to do. Things like did I want the full block transfer to always start at the beginning of the transfer block, or at an arbitrary sector offset? Did I want the single-sector transfer to transfer to and from a single-sector buffer, or to and from an offset within the full-block transfer buffer? And where did I want the DynaStat functions to place their transfer sample data?

"The point was I was developing an abstract function to do whatever SpinRite needed. But unlike the system's underlying IO that is generic and doesn't know anything about SpinRite, I could design these abstract functions, or actually this one abstract function, to do exactly what SpinRite needed. But I couldn't answer those questions of what SpinRite needed until I took a close look at SpinRite's core work loop to remind myself what exactly it was doing and how. I had been avoiding doing that since I had been hoping to leave it as much as-is as possible. Of course, I knew what it was doing broadly, but I hadn't looked at it closely under the new pure linear addressing mode approach."

Because that's the big thing that's changed is that, you know, the BIOS only sees things as cylinder heads and sectors, the traditional so-called "3D addressing." There's an extension to it that kind of makes it linear, but it's very poorly supported, that didn't really happen much, mostly add-on cards that would bring a BIOS along. But the motherboard never bothered to because as long as you can boot the OS, who cares anymore? So anyway, so the point was that the group already knew that I had recognized the only way to go forward was to scrap this cylinder head-and-sector approach, which absolutely bears no relationship to reality any longer. All drives are LBA, Linear Block Addressing. NVMe, Linear Block Addressing. Anything that we're talking to, Linear Block Addressing.

So I said: "The short version is, it all had to go. Nothing about the way it was written 34 years ago still made sense today. And even though SpinRite went through five major feature upgrades, the last one was 17 years ago, half of its 34-year life ago. As we know, SpinRite's original mission was to non-destructively low-level reformat drives. It did this one track at a time. It would read all of the data from a track, really, really reading it, no matter what. Then it would low-level reformat that one track, which might cause verified defective sectors to be moved into different logical sectors and previously good logical sectors to suddenly become defective due to re-interleaving. So SpinRite untangled all of that, rewrote the track's data back to the track, then moved onto the next track, and so on until it was finished.

"Even though SpinRite has not been doing much of that for quite some time, all of that logic had remained essentially unchanged until now. SpinRite's original philosophy had never been updated. It hadn't been getting in the way, but neither has SpinRite been operating at nearly the speed and capacity that its new drivers will now enable. I suppose an analogy might be we've built powerful new jet engines, but we can't really hang them onto the balsa wood body that was sufficient when it was being powered by rubber bands. An example would be that SpinRite's track buffer was a single 64K segment. Since transfers cannot cross a segment boundary, this was an absolute limit. 64K is 128 sectors, so no transfer could be larger than 128 sectors. But now we come along with 32,768-sector transfers into 16MB buffers. There's just no way for SpinRite's existing core code to deal with that change.

"And when SpinRite was zipping along on a drive, it was doing track-by-track transfers. But if it hit any trouble on a track, since SpinRite was always track-based, it would drop out of track mode into sector mode, where it would assess each of the track's sectors one by one to determine what to do about that track. But it makes no sense, when we're transferring 32,768 sectors at a time, to 'drop into sector by sector mode' for all of the buffer's probably fine 32,768 sectors. So the new SpinRite will have 'restartable' mass block transfers where a problem sector will stall the transfer, will work on that one sector, then resume the mass transfer starting with the sector that follows.

"Anyway," I wrote, "the point is this exactly matches the evolution of our media, and there's no getting around the fact that it needed to be done. So although I have not yet rewritten the new inner core for SpinRite, I have completely specified its operation and design. So I now know exactly what the new IO abstraction layer will need to provide to it, and I'm back to work on that."

So anyway, that's my most recent project status posting to the grc.spinrite.dev newsgroup. And if it sounds like I'm pretty much rewriting SpinRite, then you have a pretty accurate sense for what I've been doing and for what was needed. There really wasn't any way to just quickly graft on the new stuff that I had promised for this no-charge v6.1 release. And the earlier discovery that SpinRite can repair solid-state media, and also our more recent discovery that it's going to be able to sense when specific regions of apparently healthy solid-state media are actually slipping into trouble means that there's a lot of life left in SpinRite.

So yes, I'm investing hugely, ridiculously, actually, in the work for a free upgrade. But I have no problem with that since the moment 6.1 is launched, I'm going to set to immediately moving SpinRite over to its new home, that pure 32-bit real-time embedded operating system that will be able to dual-boot on either BIOS or UEFI systems, then immediately add native support for USB and NVMe hardware interfaces, then work to bring those subtle solid-state read-timing anomalies we discovered into SpinRite's UI, both to display those speed variations and to then selectively rewrite them, which we have already confirmed, in a very blunt way, actually does restore their speed and almost certainly improves their long-term read-back reliability.

And if I sound excited, you're right about that, too. SpinRite has 34 years behind it so far. I suspect it's going to easily hit 40 and probably go beyond. So that's where we are.

Leo: So you just described a decade-long plan, I think, roadmap. I'm saying, I'm just saying. I'm being optimistic here. But good. I want to keep you employed, fully employed.

Steve: We do not need to, I mean, I'm having so much fun being back to it.

Leo: Yeah.

Steve: Yeah, I love to code. I settle down here. Lorrie is great about getting me out of the house in the morning. I make myself a latte. I settle down here. And I just - I feel so good. It's like after SQRL, where I felt like I was stealing time from SpinRite, now I'm finally doing what I'm supposed to be doing.

Leo: This is your real mission. Yes, I think so, yeah.

Steve: Yeah, yeah. Just it makes sense, and I just love it. So anyway, that's where the time is going. Basically everybody is going to get for free a radically brand new SpinRite.

Leo: Sounds like an all-new program, yeah.

Steve: It is. But I did promise it. And moving forward, everyone's also going to want 7, so I'm not worried about giving all this work away for free because it's just a tease, really. I mean, it is what I promised I would do, but it's also a tease to 7 because wait till you see what I'm going to make SpinRite 7 do.

Leo: Yay.

Steve: Yeah. Okay.

Leo: SQRL was aptly named, I have to say. Unconsciously so, but I think aptly named.

Steve: Okay.

Leo: That's good. No, you know what? You've got to have a hobby. That's what I'm saying.

Steve: Well, with SQRL, I mean, and I thought about this. I couldn't very well drop it when it was half done.

Leo: No, no.

Steve: Because then we'd have nothing.

Leo: Thank god you're persistent. Absolutely.

Steve: I never quit.

Leo: Yeah. That's really a - I quit everything. So I really admire your stick-to-it-iveness. I have yet to finish a coding project, not one.

Steve: SpinRite has been the miracle of my life. I have wondered, like, what would I have - I'm sure I would have come up with something. But I don't know what it would have been. And I'm glad...

Leo: It's perfect. No, this is...

Steve: ...it was SpinRite.

Leo: You were living your absolute best life. This is absolutely what you should be doing. There's just no doubt about it.

Steve: Because it combines my love of hardware and software.

Leo: It's perfect.

Steve: And finding good solutions and so forth.

Leo: At this point I doubt anybody understands how hard drives work better than you do. I mean, you really - mass storage, pardon me, because you've also done the SSD stuff.

Steve: I'm sure that the engineers that are developing this stuff do.

Leo: Yeah, but they're narrow. See, you have to know the entire range of possibilities.

Steve: Well, a perfect example is that, although it was painful, I have one AHCI driver that runs on everything.

Leo: Right. Perfect.

Steve: Everybody else has AHCI drivers.

Leo: Right, specifically, yeah.

Steve: I don't need them. I solved the one problem that runs on everything. And again, so that's the way I work is it's harder, but we end up with something that has incredible longevity. I mean, SpinRite 6, 17 years. And SpinRite itself, 34.

Leo: It's amazing. It's amazing, yeah.

Steve: So anyway, and it doesn't look like it's over yet, either.

Leo: On behalf of all of us, thank you, Steve.

Steve: Thank you. Well, okay. We're going to see a transformation in ransomware. I talked about it in the first half. Here's some interesting insider information. So a month ago, while it was still running hard, Sophos, as part of their new "what to expect" series, posted: "What to expect when you've been hit with Avaddon ransomware." And they wrote - this is Sophos: "Avaddon ransomware is a Ransomware as a Service that combines encryption with data theft and extortion. Avaddon has been around since 2019, but has become more prominent and aggressive since June of 2020." Okay, so think about that. One year. June of 2020. "Affiliates or customers of the service have been observed deploying Avaddon to a wide range of targets in multiple countries, often through malicious spam and phishing campaigns that carry booby-trapped JavaScript files.

"Organizations hit with Avaddon ransomware face more than just data encryption. There is also the threat of public data exposure on the Avaddon leak site and, more recently, the risk of distributed denial of service attacks disrupting operations. These tactics are designed to increase pressure on victims to the ransom demand. The following information may help IT admins facing the impact of an attack with Avaddon ransomware." And they added in italics: "According to reports appearing from May 17th, 2021" - so just the previous month - "the operators behind Avaddon ransomware have taken the service 'private,' possibly by being more selective about affiliates and their targets; and they have said we will not support attacks on sectors such as government, healthcare, educational, and charity organizations." So that's interesting.

Last month, Avaddon was viewed as a real threat on the RaaS landscape. In fact, Malwarebytes wrote: "If you may recall, Avaddon is a big game hunting" - so now we have the abbreviation again, BGH - "ransomware as a service (RaaS) tool that the U.S. Federal Bureau of Investigation (FBI) and the Australian Cyber Security Centre (ACSC) warned organizations about last month." Malwarebytes wrote: "While various sectors of Australia were noted to be particularly targeted, the Avaddon strain has been instrumental in the successful network compromise of the Asian division of the AXA Group, one of the biggest cyber insurance companies in the world. Avaddon threat actors were able to extract information about what appears to be client info - passports, bank account information, ID cards, contracts, fraud-related hospital files, and other medical reports containing sensitive data about patients, and more.

"Coincidentally, this attack came close to a week after the insurance giant announced that it would cease covering customers in France who pay up after being attacked by ransomware. An insurance company refusing to cover for any monetary loss over a cyberattack will no doubt significantly increase the likelihood of victim companies refusing to cough up money to ransomware gangs." And they finish: "Schepisi Communication, an Australia-based telecom service provider, was also hit by Avaddon last month after its platinum partner, Telstra, fell victim to a ransomware attack by the same group. The criminals claimed to have access to data of a large amount of SIM cards, mobile devices, contracts, and banking information, to name a few. When the company refused to pay the demand, their official website was downed by distributed denial of service attacks, taking their website offline for several days."

Okay. So according to the FBI, Avaddon ransomware actors have compromised victims through remote access login credentials such as Remote Desktop Protocol and Private Virtual Network credentials. After Avaddon actors gain access to a victim's network, they map the network and identify backups for deletion and/or encryption. The malware escalates its privileges, contains anti-analysis protection code, enables persistence on a victim system, and verifies the victim is not located in Commonwealth of Independent States (CIS) countries. Finally, a copy of the victim's data is exfiltrated before the victim's systems are encrypted.

So I thought that Malwarebytes' use of the big game hunting (BGH) was interesting, especially in light of the fact that these guys have made what was probably the same mistake that the DarkSide gang made. They became too high-profile. Their attempt to take their operation private last month was likely their means of hoping to regain control over their out-of-control affiliates, who were attacking irresponsibly, or at least without apparent regard for social responsibility which, lo and behold, as I noted earlier, in the wake of DarkSide and JBS, has suddenly become a thing that ransomware attackers need to consider.

And then Friday before last, on the 11th, we received some surprising news from BleepingComputer, who had themselves received a surprise gift. Lawrence Abrams, the founder of BleepingComputer, wrote: "The Avaddon ransomware gang has shut down operation and released the decryption keys for their victims to BleepingComputer. This morning, BleepingComputer received an anonymous tip pretending to be from the FBI that contained a password and a link to a password-protected ZIP file. This file claimed to be the 'Decryption Keys Ransomware Avaddon' and contained three files. After sharing the files with Fabian Wosar of Emsisoft and Michael Gillespie of Coveware, they confirmed that the keys are legitimate. Using a test decryptor shared with BleepingComputer by Emsisoft, I," wrote Lawrence Abrams, "decrypted a virtual machine encrypted with a recent sample of Avaddon.

"In total," he wrote, "the threat actors sent us 2,934 decryption keys, where each key corresponds to a specific victim. Emsisoft has released a free decryptor that all victims can use to recover their files for free. While it doesn't happen often enough," he wrote, "ransomware groups have previously released decryption keys to BleepingComputer and other researchers as a gesture of goodwill when they shut down or release a new version.

"Over time, Avaddon has grown into one of the larger ransomware operations, with the FBI and Australian law enforcement recently releasing advisories related to the group." And I think that may be the key. "At this time, all of Avaddon's Tor sites are inaccessible, indicating that the ransomware operation has likely shut down. Furthermore, ransomware negotiation firms and incident responders saw a mad rush by Avaddon over the past few days to finalize ransom payments from existing unpaid victims." To me that sounds like they were given a deadline.

So he finishes: "Coveware CEO Bill Siegel has told BleepingComputer that Avaddon's average ransom demand was around 600K. However, over the past few days, Avaddon has been pressuring victims to pay, and accepting the last counteroffer without any pushback, which Siegel states is abnormal." And he finishes: "It's not clear why Avaddon shut down, but it was likely caused by the increased pressure and scrutiny by law enforcement and governments worldwide after recent attacks against critical infrastructure." Emsisoft's threat analyst Brett Callow told BleepingComputer: "The recent actions by law enforcement have made some threat actors nervous. This is the result. One down, and let's hope some others go down, too."

And this brings us to the view from Russia, with a lot of interesting inside information, brought to us by two Russians, Vitali Kremez and Yelisey Boguslavskiy. Together they run AdvIntel a contraction of Advanced Intelligence, A-D-V-I-N-T-E-L and they offer unique insight thanks to having access to unique data and players in their home country. They explain why it all comes down to one thing, and why I titled this podcast "Avaddon Ransonomics." So I've edited and tweaked their write-up actually in some cases significantly to fix some minor Russian-as-their-first-language errors and to clarify things here and there. But it's essentially untouched.

They begin by explaining the ransomware gang's name: "The three-letter Hebrew root 'avad'" - and that's four letters, but okay - "from which the name Avaddon is derived has two main semantic interpretations, 'to destroy' and 'to lose or get lost.' Indeed," they wrote, "these two meanings perfectly define the Avaddon ransomware, a destructive and malicious force which always managed to conceal and disappear."

They wrote: "Today we shed light on this lost and hidden criminal empire using unique datasets, the full list of Avaddon victims ever targeted by the group over the year of its existence, discovered by AdvIntel. This unique SIGINT data is supported by exclusive HUMINT findings, statements made by the Eastern-European underground cyber community leaders who worked with Avaddon, explaining and interpreting the group's rapid rise and even more rapid downfall.

"On June 11th, 2021, Avaddon released keys for over 2,000 victims containing the exact company breach names. Our analysis of the confirmed victimology shows that some of them are the world's leading companies. How did this group succeed in hitting so many companies within a year? The answer is Avaddon created an entire ecosystem around themselves, a web of supply chains, international affiliates, sellers, underground auction managers, and negotiators. They have established an organic ecosystem of criminal extortion economy, a form of 'ransonomics.'

"Of course, Avaddon was not the only group pursuing a diversified approach to building a larger business system. However, they were likely the most creative ones. They were the only Russian-speaking group that enabled, but promoted, international partners joining the team as affiliates that directly represented the coverage of Avaddon's attacks, reaching five continents. One of Avaddon's largest attacks on a major financial institution occurred in May of 2021." And of course that's the attack on AXA. "It illustrates this integrated approach of building the ransomware-attacks economy.

"While investigating the AXA attack," they wrote, "we discovered 141 unique indicators for RDP compromises for the victim's domain. This means that Avaddon was using the services of an RDP brute-forcing group. Moreover, two weeks before the attack, a threat actor conveniently published a post on a major underground forum where Avaddon was based, auctioning classified information on the future victim. This access seller happened to be connected to a malware developer specializing in data exfiltration tools. In other words, before Avaddon performed their data-stealing operation, they were able to utilize the entirety of underground services and purchase the full set - RDP access, direct network access, and malware for data exfiltration.

"This innovative approach enabled Avaddon to perform several thousand attacks. AdvIntel has analyzed Avaddon's victims' unique datasets to build the most definitive adversarial profile. Traditionally, while profiling the group's victimology, companies rely on the data available in public, i.e., ransomware websites. And indeed, even looking at this partial data, which only includes companies whose information was dumped on the shame blogs, we can see that Avaddon played a major role in the threat landscape.

"However, the victims whose names were published on the shame blog are only the tip of the iceberg. AdvIntel's advanced dataset, covering all Avaddon victims, provides further visibility into the gang's operations. For this statistical research, AdvIntel has selected a special high-value target dataset. First, we defined the industries which were the primary targets for the group, manufacturing, retail, technology, and engineering being the most preferred sectors most likely because for the companies of these sectors even a brief interruption of business can imply fatal consequences.

"For the next step, we performed market research of the victims' revenue to identify the potential pattern of Avaddon attacks. The total revenue of all victims was around" - and this is aggregate total earnings over their life - "35 billion USD. This is the segment of the market which has been in one way or another threatened by Avaddon's malicious operations. Avaddon's victims can be divided into three categories - small, medium, and large. The average per-victim lifetime revenue was: 13 million USD for small businesses; 287 million USD for medium-sized businesses; and 3.7 billion USD for large businesses."

They said: "Our next research goal was to calculate how much money the Avaddon group could make before their rapid retirement. We have utilized our previous knowledge from threat actor engagements to develop realistic formulas of ransom demand calculations supported by the actual Avaddon cases. Traditionally, all Russian-speaking actors are using the victim's annual revenue to calculate the ransom. After identifying the revenue, they investigate the sector within which the victim operates.

"The most common calculation which according to our sensitive and credible source intelligence as used by Avaddon was the so-called '5x5' rule where 5% of the target's annual revenue is used to start the negotiations, with annual revenue estimated as one-fifth of the total historical revenue. In other words, for a victim which has a total lifetime revenue of 7 million USD, the starting ransom price will be 70,000 USD. Typically, Avaddon dropped the price during bargaining, and the end ransom was around 50,000 USD for a successful operation.

"However, not all companies out of the 2,000 victim list were forced to pay such ransom. In many cases, the negotiation failed or the ransom was minimal, several thousand USD, especially in the beginning. At the same time, bigger payments were demanded from larger entities. Here, the '5x5' formula would be replaced by a more tempered scale for larger ransom involving 0.01% margins for annual revenue instead of 5%, et cetera. So for a multi-billion dollar company, the demand was constrained to a few million dollars.

"After finalizing the calculations with a case-by-case study of each victim from the high-value dataset, AdvIntel assessed that the bulk of ransom payments came from over a thousand smaller-sized companies, from which was demanded between $30,000 to $70,000, and constituted the overall aggregate payment of 55 million over its lifetime to Avaddon. Over the 500 larger businesses in the victim list, that constituted another 30 million, and the rest was divided between smaller payments. Our assessment of Avaddon's lifetime, approximately one year, income is therefore approximately 87 million USD.

"Our team has also attempted to calculate the revenue of a core Avaddon team member based on these numbers. Within Avaddon RaaS, over 70% of income went to affiliates. Therefore, the core team, and especially the leader of Avaddon, received around $26 million. This number was likely divided between at least four individuals, which made the approximate annual income, that is, for a year, 7 million USD. For comparison, the median annual income in Russia is approximately 7,000 USD. In other words, in one year of ransomware development, an Avaddon member made the same money as an average Russian would make in 1,000 years." They wrote: "This is the best illustration of how lucrative ransomware could be for the region."

So they finish, at length: "If Avaddon was so successful, what could have motivated them to quit? The likely answer is fear. U.S. law enforcement and the Biden administration became very upfront regarding future retaliatory measures against ransomware and the new angle in which ransomware is seen as essentially an act of terrorism." And remember, this is from two Russians. "This new take," they write, "on digital extortion from the world's leading superpower had a direct effect within the underground community. The above-mentioned ransonomics, which powered a carefully and meticulously built web of alliances and supply chains, began to rapidly fail. Software brokers refused to sell malware to ransomware groups, forums banned Ransomware as a Service partnerships, and affiliates were left without means and services to disseminate the payload.

"The cybercrime world has always been similar to piracy, and it has its own black mark. But after the Colonial Pipeline incidents, ransomware was clearly carrying the same black mark for the first time. Avaddon, which was in the center of the dynamic and turbulent ransomware ecosystem, quickly realized the risks they might face. This realization was likely caused by the recent intervention of politics into the cybercrime domain. Overall, the inner logic of the Russian security landscape presumes that a successful cyber group will eventually become prominent enough to attract the state's attention. Usually, law enforcement will turn a blind eye to cyber operations unless these operations target Russian citizens or businesses. However, this status quo changed in May of 2021.

"After the admin of XSS, the largest dark web forum called for a ransomware ban, justifying it for political reasons. The community of digital extortionists in Russia was observed to go through stages of paranoia. This was also the result of multiple statements made in the last three months by the Russian government, the Russian Ministry of Foreign Affairs, and by President Putin personally about establishing an international Russian-American initiative to establish a joint cybersecurity landscape. The Russian officials likely see this as a tool of deescalating the U.S.-Russian relationships, especially in light of the upcoming Biden/Putin summit scheduled for June 16th, 2021.

"Indeed," they wrote, "the Russian government traditionally goes through rounds of escalation and deescalation with the West. The escalation phase involving military maneuvers in the proximity of the Russia-Ukraine border and in Northern Syria ended in April of 2021. Now the Kremlin, aiming to address severe challenges in the post-COVID economic recession and the turbulent domestic situation, is interested in creating a certain framework of stability in the international arena and ensuring stabilized relationships with the U.S. to avoid unnecessary pressure. Therefore, cybersecurity, a controversial issue for the U.S.-Russia relationship, is on the frontlines of this deescalation agenda.

"It is also noteworthy that some of the jurisdictions that were targeted by Avaddon - Iran, China, and Turkey - also have strong geopolitical ties with Russia and act as Russian allies or critical economic partners. However, it's unclear if this could have led to any aggravation in the relationship between Avaddon and the Russian state. Whatever the true rationale of the Russian politicians calling for international cybersecurity cooperation is, these recent statements have clearly had an impact on the underground cybercrime community. AdvIntel has tracked multiple discussions between top-tier actors working with Avaddon who mentioned that one of the group's affiliates was apprehended by the Russian law enforcement on the eve of the U.S.-Russia summit, and that further arrests may follow against the ransomware leaders in order to secure the political landscape."

So to me it seems very clear that they were in a big hurry, Avaddon was, to shut themselves down. Thus that weird behavior that was seen in the final days of Avaddon, basically taking anything they could get from the remaining victims and then finally releasing all the keys because for whatever reason they were going to be able - like for some reason they were going to be able to never get anymore ransoms paid again. And so they scraped up the residuals that they could. And then because they knew for whatever reason they were not going to get anymore ransom payments, they just released the keys for free.

Leo: Right. Very interesting.

Steve: Yeah. And, you know, this is not U.S. propaganda. This is two Russians with a Russian intelligence firm who have contacts, who know the affiliates of Avaddon, who are saying it's gotten too hot. We can't do this.

Leo: Putin says, "You cut it out for a little bit, okay? We'll get you later." Very, very good stuff, as usual, Steve. I really enjoyed the update, too, on the SpinRite. Can't wait to see some more stuff from you. It's exciting.

Steve: Me, too. I'm going to go working on it this evening.

Leo: Well, if you want to know more, if you want to get a copy of SpinRite and get the free upgrade, all you have to do is go to GRC.com. Forums.grc.com you can read up on it, keep up with what Steve's doing because he does post regularly to the forums. And also while you're there you can get a copy of the show. Steve's got a couple of unique formats. He's got the 64Kb audio like we do, but he also has a 16Kb audio, very small audio file. It's a little scratchier, but if you don't have lot of bandwidth, it might save you some download speeds, some download time. He also has the human-written transcripts. Elaine Farris does a great job with those so you can read along as you listen. That's all at GRC.com. You can leave feedback for Steve at GRC.com/feedback, or on his Twitter account. His DMs are open. His Twitter handle is @SGgrc.

We have 64Kb audio and video, as well, on our website, TWiT.tv/sn. We do the show Tuesdays, right after MacBreak Weekly, so that's usually around 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you want to tune in and watch it live, there's a live audio stream and video stream at TWiT.tv/live. People who watch live often like to chat with others who are watching live. Couple of places to do that. Our IRC, of course, is open at irc.twit.tv.

What else? There's a YouTube channel with all the videos there. Actually, if you go to TWiT.tv/sn, you'll find a link to the YouTube. You'll find a direct link to various podcast players, but also an RSS link you can add to any podcast player. Really subscribing is probably the easiest way to make sure you get your weekly fix of Security Now!. And if your podcast program has a review section, do us a favor, leave a five-star review for Steve. I think he earns that every single week. Absolutely.

Thank you, Steve. Have a great week.

Steve: Thanks, buddy.

Leo: See you next week on Security Now!.

Steve: Bye.