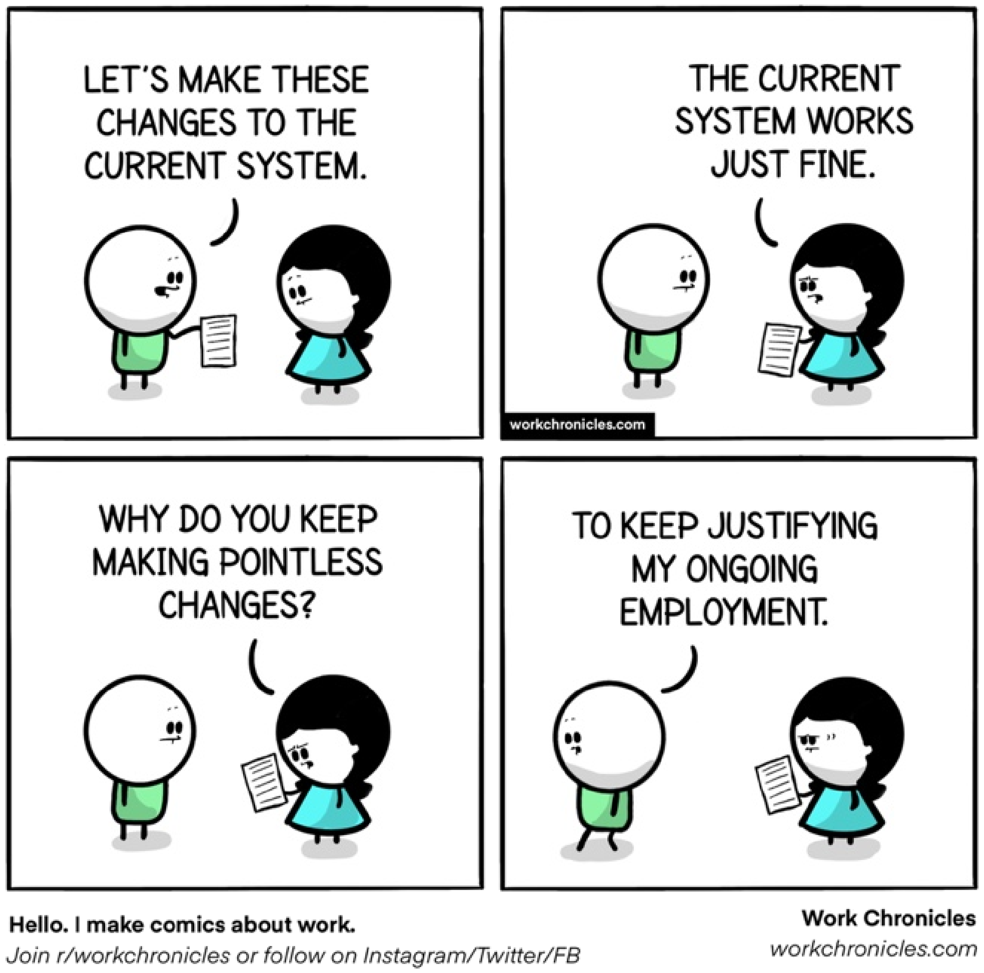
Security Now! #824 - 06-22-21
Avaddon Ransonomics

This week on Security Now!

This week, believe it or not, we have yet another 0-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 SHA256 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. 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.

"Windows 11"


LET'S MAKE THESE CHANGES TO THE CURRENT SYSTEM.
THE CURRENT SYSTEM WORKS JUST FINE.
workchronicles.com
WHY DO YOU KEEP MAKING POINTLESS CHANGES?
TO KEEP JUSTIFYING MY ONGOING EMPLOYMENT.
Hello. I make comics about work.
Join r/workchronicles or follow on Instagram/Twitter/FB
Work Chronicles
workchronicles.com

Web Browser News

Another day, another Chrome 0-day

As I said last week, this is what it's like to be the world's #1 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 7th actively exploited in the wild 0-day that the Chromium team has patched so far this year. We're now at 91.0.4472.114 for the three desktops. It resolves 4 security vulnerabilities, including that -30554 which was a high severity, use-after-free vulnerability in WebGL — the Web Graphics Library — which is a JavaScript API used to for rendering interactive 2D and 3D graphics in the browser.

As it's formally described, 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. But we know that the bad guys are much less interested in corrupting some data or crashing the user's browser than they are in executing something of their choosing own. Since this vulnerability was being actively exploited in the wild, in this instance we can ignore the lesser possibilities and assume that remote code execution had been achieved and was being exploited.

If the CVE number 30554 sounds familiar, that may be because 30551 was the previous 0-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, 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.

- CVE-2021-21148 - February 4th, 2021
- CVE-2021-21166 - March 2nd, 2021
- CVE-2021-21193 - March 12th, 2021
- CVE-2021-21220 - April 13th, 2021
- CVE-2021-21224 - April 20th, 2021
- CVE-2021-30551 - June 9th, 2021

Shane Huntley, Director of Google's Threat Analysis Group, Tweeted two weeks ago, on June 8:

"I'm happy we are 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."

https://twitter.com/ShaneHuntley

I read through Shane's surrounding Tweets in an attempt to understand what he meant by his concern over "the role of commercial providers." Leo: Any idea what he might mean by that??

Ransomware

Ransomware perpetrators are increasingly purchasing access
The security firm ProofPoint has been tracking the ransomware underground for many years. 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.

This is the trend that we've talked about previously. This was the way we believe the Colonial Pipeline attack began. An existing VPN logon credential was purchased on the dark web by a DarkSide affiliate and used to gain entry into Colonial Pipeline's internal network. ProofPoint's research confirms this trend, and puts a little more meat on the bone. They explain that:

*Ransomware threat actors currently carry out "big game hunting," conducting open-source surveillance to identify high-value organizations, susceptible targets, and companies' likely willingness to pay a ransom. Working with **initial access brokers**, ransomware threat actors can leverage existing malware backdoors to enable lateral movement and full domain compromise before successful encryption. An attack chain leveraging initial access brokers could look like this:*

1. *A threat actor sends emails containing a malicious Office document*
2. *A user downloads the document and enables macros which drops a malware payload*
3. *The actor leverages the backdoor access to exfiltrate system information*
4. ***At this point, the initial access broker can sell access to another threat actor***
5. *The actor deploys Cobalt Strike via the malware backdoor access which enables lateral movement within the network*
6. *The actor obtains full domain compromise via Active Directory*
7. *The actor deploys ransomware to all domain-joined workstations*

Just like an organic virus which mutates to improve its chances of survival, we see here a similar mechanism in 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 dark web escrow services formalized as uninterested 3rd parties to manage and apportion ransom payments. And now we're seeing the emergence of "IAB's" — Initial Access Brokers — as the ransomware affiliate role divides and specializes into initial entry and post-entry exploitation functions.

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 network border device and would typically set up a Bot in a router that would contact a Command & 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 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.”

Attempting to hold infrastructure at ransom, while it may appear at first to be the motherlode, 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 just as much a mistake. Sure, they netted $11 million dollars, but they also got the U.S. to start considering ransomware as 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 eleven $1 million dollar 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 3rd-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. 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.

Remember back a few years ago when we were talking about how the networks of managed service providers were being compromised; and their clients, 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 attacks never became big news. We watched them and were aware, 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.

Security News

A weird bug in iOS Wi-Fi

An interesting and somewhat humorous bug has been uncovered by iOS's parsing of network SSID's.

There has long been a concept in programming languages, at least since FORTRAN (where I recall it being there), of using a so-called format string to describe the shape of the contents of input coming into a computer, or the way some output variables should be presented. So, for example, an output greeting might be formatted: “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 follow 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. %d, for example, 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 at iOS somehow discovered that if a Wi-Fi network's SSID's name was set to: "%p%s%s%s%s%n" and an iOS device then attempted to join any Wi-Fi network having that name, the device's Wi-Fi would become immediately and semi-permanently inoperative. A restart/reboot would have no effect and all logical attempts to reverse the change would fail. Any attempt to enable the Wi-Fi subsystem to fix the trouble would immediately crash it before the user could use it to resolve the problem.

This is one of those things that's sort of our collective fault 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' 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 have been happening is that the SSID string which, itself, looked like escape-formatted text confused some formatting parser somewhere and likely caused the internal Wi-Fi system to crash... and crash and crash and crash.

The concern was that as the news of this spread, annoying jerks would immediately begin exploiting this discovery. And, indeed, that happened. Postings began appearing telling naive users that they could obtain much faster WiFi by renaming their access points accordingly. (Shhhhh!! Don't tell anyone!)

Fortunately, after some additional experimenting it was discovered that the device's WiFi function could be restored by going to: Settings > General > Reset > Reset Network Settings. That would flush out the existing problematical setting.

So... How did we get into this trouble in the first place? We would really have to say that this is 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 into a single stream is a recipe for disaster. So why do we do it? We do it because it's **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 {name}={value} pairs separated by '&' ampersands. But how do you have a name or a value containing an '=' equals 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, indeed, countless mistakes **have** been made since the beginning of the web by programmers who were not sufficiently concerned and cautious. And while I'm certain that the details with iOS's WiFi are different, and that the problem will be trivial to repair, it does appear that this problem just bit the Apple.

Miscellany

Windows 11

Paul Thurrott, I, many of those in GRC's newsgroups and most of the rest of the world 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 they're not today as this has spread far and wide. 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 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 offer our inquisitive listeners some protection in the form of the SHA256 hash of the known-to-be-valid .ISO file. That way, anyone here who believes they may have obtained the ISO can readily verify its validity and safety. Being very cautious myself, last week I downloaded many of these 4.8 gigabyte apparent Win11 .ISOs from very different sources and checked the SHA256 hash of each. They all matched:

b8426650c24a765c24083597a1eba48d9164802bd273b678c4fefe2a6da60dcb

The SHA256 hash is in the show notes. Its first 32 bits are "b8 42 66 50" and the last 32 bits are "6d a6 0d cb". Those 64 bits alone provide 1 in 18.4 x 10^18 verification strength. So if your hash begins and ends with those, you're okay.

There are various Explorer shell file-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:

Standard Command Prompt:	certutil -hashfile "Windows-11-Dev-OS-21996.1.iso" SHA256
Powershell's built-in function:	Get-FileHash .\Windows-11-Dev-OS-21996.1.iso

Having played around with it, it is indeed Windows 11. And I have to say, it is truly gorgeous. It is VERY pretty. But Leo, I'm a bit disappointed... because I keep encountering plenty of pointy corners. 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 to 150%. For whatever reason, what I'm seeing still seems quite pointy in places.

There are reports of Win11 install failures on older machines, presumably primarily older desktop machines, which have been running Windows 10 without trouble, but on which Win11 refuses to run due to the desktop not having a TPM Trusted Platform Module, and thus being unable to support Windows Secure Boot. Support for “Secure Boot” has been optional for Win10, but for some reason Microsoft has decided to make it mandatory for Win11.

Anyone running Windows can check for the presence of their hardware platform’s TPM by running the “tpm.msc” snap-in. Just hit Windows+R to open the Run... dialog, then enter tpm.msc and hit enter to launch the TPM configuration applet.

The good news is, workarounds for the lack of TPM have already been found and are widely circulating on the Net. I’ve included a couple of links in the show notes:

https://fossbytes.com/solve-tpm-2-0-error-installing-windows-11-fixed/
https://allthings.how/how-to-fix-this-pc-cant-run-windows-11-error/

Offline Accounts: The other potential gotcha hit people who do not wish to login to Microsoft during installation and who would like to create an offline login account. Although Win11 Pro allows for bypassing this in the setup UI, the Home edition of Win11 does not. So if Win11 is being installed on hardware containing a built-in Win10 Home OEM key, it will insist upon the creation of an online account. Win11 Pro offers the “I don’t have Internet” option to bypass the establishment of a connection, but that’s been deliberately removed from the options for the Home edition.

Fortunately there’s a surprising workaround for that, too. It was discovered by Adam, whose moniker is “Warwagon” in the grc.securitynow newsgroup. It turns out that it’s possible to simply close the insistent “Let’s connect you to the Internet” dialog by hitting the Alt+F4 key combination. Adam shared his discovery with the guys at Neowin and they wrote the following:

“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 OOBE [out of box experience] process, especially in these early days when most installs of the OS are happening on virtual machines.”

https://www.neowin.net/news/windows-11-home-requires-internet-to-complete-setup-but-there039s-a-workaround/

Want the Win10 Start menu under Win11?
Those who are not fans of the new positioning and look of the Win11 Start Menu may reversibly return to more familiar territory by using the “Start_ShowClassicMode” Registry key and value:

At: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced
Start_ShowClassicMode will be a REG_DWORD defaulting to ‘0’. Set it to ‘1’ and reboot.

Closing The Loop

The Security Now! Podcast has found a new purpose...
Lost World / @Bruin144

Security Now (Just the sound of the Podcast by itself removes intruders) for the 2nd time 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. This week a new racoon defeated my security measures and moved in. 18hrs of Security Now later he moved on/out and I repaired the mesh he had pulled down.

SpinRite

I received a Twitter DM last week where one of our listeners asked *"What the hell's going on with SpinRite and when are we going to get it?"* That reasonable question reminded me that everyone in GRC's spinrite.dev newsgroup knows exactly what's going on and where things stand. But that's a small fraction of the people we have listening to this podcast, and since everyone has local mass storage that they care about, 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 sense for what is happening while they are being patient. And because this v6.1 project has also grown into so much more than I thought, 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. So the last test was enumerating all drives and showing all of each drive's relevant data **far** more comprehensively than any previous release of SpinRite. That release went amazingly well. Actually much better than I expected, probably because we've been moving forward carefully and only leave fully tested and verified code in our wake.

After that, the next thing I planned was to bring the drive benchmarking system online since SpinRite's built-in read performance benchmarking would be testing the read-channel for every drive.

Exactly one month ago, on May 22nd, I posted an update under the subject "One thing leads to another." So I want to share that post and the four subsequent posts I've made 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 snapshot interesting...

One thing leads to another… (May 22nd)
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 next **beyond** v6.1. That's coloring many of the decisions I'm making along the way. So I'm not really writing just for this, but also to a very large degree for 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, and sometimes it's able to use a 32-bit linear buffer in high memory. 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 v6.1 will be running in flat real mode.

So each drive see by SpinRite can be one of five different access classes:

1. BIOS only. (a)(d)
2. Extended BIOS without direct hardware access details. (b)(e)
3. Extended BIOS =with= direct hardware access details. (c)(e)
4. An IDE or SATA drive on PCI with Bus Mastering. (c)(f)
5. A SATA drive on PCI with an AHCI controller. (c)(f)

And, as shown with the note letters above, several of those classes have sub-class access feature variations:

a: Limited to max CHS size (28 bits): 137 GB.
b: Potentially any size drive.
c: Access to the entire drive plus SMART & Log data.
d: Segmented memory transfers only.
e: Possible use of large sector count transfers and linear RAM.
f: 16MB (32k sector) transfers to high memory.

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, 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 "in 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 (single sector or entire track) 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, where the location of the buffers is similarly abstracted.

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 bypassed for a small subset of special case accesses. 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 operations 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.

Change in logging operation?? (May 29th)
(Explain about a session count trimmed log located in each drive's root dir.

SRv6.1 Progress Report (June 9th)
Gang...

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."

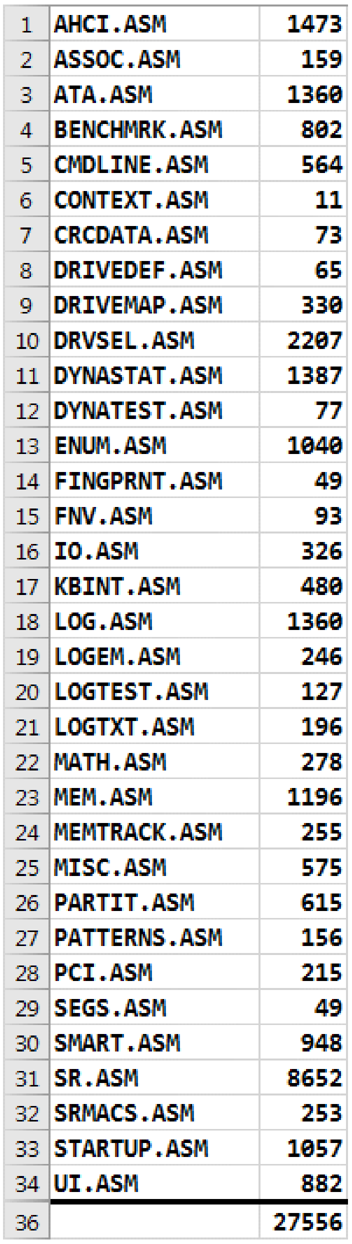
In order to catch every instance of something that needs changing, I change the name of a variable to force assembly errors due to the old name no longer existing. That's necessary because the old variable might have been 16 bits and the new one needs to be 64 bits. But it means that a daunting list of errors results from every reference to the now-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 the previous "OperatingLocationLow" and "OperatingLocationHigh" variables. They were each originally 16 bits back in the pre 32-bit days. So I needed both a "low" 16 bits and a "high" 16 bits to get to 32 bits. They are being replaced by a single 64-bit "TransferLocation" variable. 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 the welcome surprise that there are no more errors because all references to the new variables will have been encountered and re-coded.

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 that 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.



1	AHCI.ASM	1473
2	ASSOC.ASM	159
3	ATA.ASM	1360
4	BENCHMRK.ASM	802
5	CMDLINE.ASM	564
6	CONTEXT.ASM	11
7	CRCDATA.ASM	73
8	DRIVEDEF.ASM	65
9	DRIVEMAP.ASM	330
10	DRVSEL.ASM	2207
11	DYNASTAT.ASM	1387
12	DYNATEST.ASM	77
13	ENUM.ASM	1040
14	FINGPRNT.ASM	49
15	FNV.ASM	93
16	IO.ASM	326
17	KBINT.ASM	480
18	LOG.ASM	1360
19	LOGEM.ASM	246
20	LOGTEST.ASM	127
21	LOGTXT.ASM	196
22	MATH.ASM	278
23	MEM.ASM	1196
24	MEMTRACK.ASM	255
25	MISC.ASM	575
26	PARTIT.ASM	615
27	PATTERNS.ASM	156
28	PCI.ASM	215
29	SEGS.ASM	49
30	SMART.ASM	948
31	SR.ASM	8652
32	SRMACS.ASM	253
33	STARTUP.ASM	1057
34	UI.ASM	882
36		27556

SpinRite's source code line counts (June 11th — two days later)
Gang,

During my after-dinner walk with Lorrie, I mentioned that once things had stabilized and settled down with SpinRite, 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:

https://www.grc.com/dev/spinrite/SpinRite-Source-Line-Counts.png

Yep... 27,556 lines. That's not all code, since 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 make clear. But it does give some sense for SpinRite's source code base.

I excluded a handful of files of UI content: the screens, screen composition definitions and text that fills them, since they aren't code.

In the beginning, SpinRite was mostly a single SR.ASM file. And you can still see that heritage since SR.ASM remains, by far, the largest single file at 8652 lines. I've been breaking it apart into smaller more manageable/functional pieces during this recent work. So things like MATH.ASM and MEM.ASM were once part of SR.ASM but are now separate files. I broke them out since they needed lots of reworking and rewriting.

I won't need to re-read any of what I have written recently, like AHCI.ASM, ASSOC.ASM, ATA.ASM, DRIVEMAP.ASM, DRVSEL.ASM, ENUM.ASM, FNV.ASM, IO.ASM, MATH.ASM, MEM.ASM, PCI.ASM and UI.ASM, which have all been written from scratch during this recent round of work.

But I should read SR.ASM and STARTUP.ASM from top to bottom just to make sure I've re-encountered everything.

Another Update (And finally my most recent note from last Thursday, June 17th)
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 with exactly what 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/from a single-sector buffer, or to/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 designing abstract functions 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 to do exactly WHAT SpinRite needed.

But I couldn't answer those questions until I took a close look at SpinRite's core work loop to remind myself what it was doing and exactly 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.

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.) So SpinRite untangled all of that, re-wrote 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 enable. I suppose an analogy might be that 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 has always been 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, we'll work on that one sector, then resume the mass transfer with the next sector.

Anyway, 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 that new "inner core" for SpinRite, I HAVE completely specified its operation and design. So I know exactly what the new IO abstraction layer will need to provide to it and I'm back to work on that.

Okay. So that's my most recent project status postings to the grc.spinrite.dev newsgroup.

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 v6.1 is launched, I'm going to set about immediately moving SpinRite over to its new home, a 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 re-write them which we have already confirmed, in a very blunt way, actually does restore their speed and almost certainly improve their long-term read-back reliability.

And if I sound excited, you'd be right about that too. SpinRite has 34 years behind it so far. I suspect that it's going to easily hit 40 and beyond!

Avaddon Ransonomics

"The Rise & Demise of Multi-Million Ransomware Business Empire"

https://www.advanced-intel.com/post/the-rise-demise-of-multi-million-ransomware-business-empire

A month ago, Sophos, as part of their new "What to expect" series, posted: "What to expect when you've been hit with Avaddon ransomware" They wrote:

Avaddon ransomware is a Ransomware-as-a-Service (RaaS) that combines encryption with data theft and extortion. Avaddon has been around since 2019 but has become more prominent and aggressive since June 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 (DDoS) attacks disrupting operations. These tactics are designed to increase pressure on victims to the ransom demanded.

The following information may help IT admins facing the impact of an attack with Avaddon ransomware.

And then they added in italics: "According to reports appearing from May 17, 2021, the operators behind Avaddon ransomware have taken the service '"private" – possibly by being more selective about affiliates and their targets – and have said they will not support attacks on sectors such as government, healthcare, educational, and charity organizations."

So, last month, Avaddon was viewed as a real threat on the RaaS landscape. In fact,

MalwareBytes Labs wrote:

*If you may recall, Avaddon is a **big game hunting (BGH)**, ransomware-as-a-service (RaaS) tool that the US Federal Bureau of Investigation (FBI) and the Australian Cyber Security Centre (ACSC) warned organizations about last month.*

While various sectors in 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.

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 (DDoS) attacks, taking their website offline for several days.

According to the FBI, Avaddon ransomware actors have compromised victims through remote access login credentials—such as Remote Desktop Protocol (RDP) and Virtual Private Networks (VPN). 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 privileges, contains anti-analysis protection code, enables persistence on a victim system, and verifies the victim is not located in the Commonwealth of Independent States (CIS). Finally, a copy of the victim's data is exfiltrated before the victim's systems are encrypted.

I thought that Malwarebytes' use of the term 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, low and behold, has 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 surprise news from BleepingComputer who had, themselves, received a surprise gift. BleepingComputer wrote:

The Avaddon ransomware gang has shut down operation and released the decryption keys for their victims to BleepingComputer.com.

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, 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. 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.

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 push back, which Siegel states is abnormal.

It is 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."

Which brings us to the view from Russia, with a lot of interesting inside information, brought to us by two Russians "Vitali Kremez" & "Yelisey Boguslavskiy." Together they run AdvIntel — a contraction of Advanced Intelligence — 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 today's podcast "Avaddon Ransonomics."

I've edited and tweaked their write-up to fix some minor Russian-as-their-first-language errors and to clarify things here and there. But they begin by explaining the ransomware gang's name:

The three-letter Hebrew root "avad", from which the name Avaddon is derived, has two main semantic interpretations- "to destroy" and "to lose / get lost". Indeed, these two meanings perfectly define the Avaddon ransomware - a destructive and malicious force - which always managed to conceal itself and disappear.

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 cybercommunity leaders who worked with Avaddon - explaining and interpreting the groups' rapid rise and even more rapid downfall.

On June 11, 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 were 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 occurring in May 2021 *[I'm sure they're talking about the attack on AXA]* - illustrates this integrated approach of building the ransomware-attacks economy.

While investigating the [again, AXA] attack, 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 victim's 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 identified the industries which were the primary targets of the group - manufacturing, retail, technology, and engineering being the most preferred sectors most likely because, for the companies in these sectors, even a brief interruption of businesses 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 $35 Billion USD. This is the segment of the market which has been in one way or another threatened by Avaddon malicious operations.

Avaddon's victims can be divided into three categories - small, medium, and large.

The average per-victim revenue was:

- $13 Million USD for small businesses
- $287 Million USD for medium-sized victims
- $3.7 Billion USD for larger businesses

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 victims' 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 a 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 the bargaining, and the end ransom was around $50,000 USD for a successful operation.

However, not all companies out of the two thousand 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%, etc. So, for a multi-billion dollar company, the demand was constrained to a few million dollars.

After finalizing all 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 USD and constituted the overall aggregate payment of $55 million to Avaddon. Over 500 larger businesses in the victim list constituted another $30 million, and the rest was divided between smaller payments. Our assessment of Avaddon's lifetime (approximately one year) income is 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 USD. This number was likely divided between at least four individuals, which made the approximate annual income (Avaddon existed for a year) $7,000,000 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 amount of money as an average Russian would make in one thousand years. This is the best illustration of how lucrative ransomware could be for the region.

If Avaddon was so successful, what could have motivated them to quit?

The likely answer... is fear. US 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. This new take 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 RaaS 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 incident, **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 de-escalating the US-Russian relationships, especially in the light of the upcoming Biden/Putin summit scheduled for June 16, 2021.

Indeed, 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 2021. Now the Kremlin, aiming to address severe challenges of 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 US to avoid unnecessary pressure. Therefore, Cybersecurity, a controversial issue for the Russia-US relationship, is on the frontlines of this de-escalation agenda.

It is also noteworthy that some of the jurisdictions that were targeted by Avaddon - Iran, China, and Turkey also have strong geo-political ties with Russia and act as Russian allies or critical economic partners. However, it is 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 US-Russia summit and that further arrests may follow against the ransomware leaders in order to secure the political landscape.