SECURITY NOW!

Transcript of Episode #827

REvil's Clever Crypto

Description: The past week has been dominated by the unimaginable mess that Microsoft has created with what have become multiple failed attempts to patch the two PrintNightmare flaws, and the continuing "Cleanup on Aisle 5" following what is widely regarded as the single most significant ransomware supply chain attack event ever. So today we first catch up on the still sadly relevant PrintNightmare from which the industry has been unable to awaken. We'll cover a few more bits of security news. Then, as planned, we'll take a deep dive into the detailed operation of the REvil/Sodinokibi malware's cryptographic design.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-827.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-827-lq.mp3

SHOW TEASE: Coming up on Security Now!, I am in for Leo Laporte this week to talk to Steve Gibson about Security [Right] Now! because that's how it works. First, we talk about Microsoft's ongoing PrintNightmare. Yes, it continues to be a nightmare; and disabling the print spooler, well, that just ruins everything. There's a lot to talk about, what you can do to fix it, and why it's still broken. Plus we talk about how Kaseya could have actually been a lot worse than it actually was - REvil's ransomware campaign didn't do the full and complete damage it could have done - before we round things out with an in-depth look at the entire encryption method from REvil. It's quite a doozy, but Steve Gibson does his best to explain while I stare at a flowchart that makes zero sense to me. It's all coming up on Security Now!.

MIKAH SARGENT: This is Security Now!, Episode 827, recorded Tuesday, July 13th, 2021: REvil's Clever Crypto.

Hello, and welcome to Security Now!. Look, when I'm here to cohost a show I tend to like to use the description page that we have with TWiT. And so I'm going to tell you about the man who coined the term "spyware" and created the first antispyware program. It's Steve Gibson of, well, Security Now! fame, and I am honored to be joined by you today, Steve.

Steve Gibson: Hey, it's great to be with you, standing in for Leo as you are, while he and his gang are surfing somewhere, I guess.

MIKAH: Yes, yeah, gallivanting in...

Steve: Catching some rays.

MIKAH: Somebody's got to do it.

Steve: So we have, surprisingly, happily, the topic that I teased and promised last week, since really nothing happened other than the two big stories that we'll be covering this week, which is the unimaginable mess that Microsoft has created with what have now become multiple failed attempts to patch the two PrintNightmare flaws. And then, of course, this continuing cleanup on aisle five following what is widely acknowledged as the single most significant ransomware supply chain event ever. So we're going to catch up on Print Nightmare and those problems; talk about a few other things that did happen.

And then, as I promised last week, we're going to talk about in detail - and yeah, some of our listeners will glaze over a little bit probably with some of the detailed crypto, but the fun is in the details - we're going to talk about REvil's Clever Crypto architecture and the flexibility that it provides. So I think another great podcast for our listeners while Leo is getting a tan.

MIKAH: Yeah, it sounds like it.

Steve: So we always have, as you know, a Picture of the Week. It's been, I think, we sort of started this thing a decade ago, and it's always fun. And as a consequence, I'm getting pictures send to me from Twitter followers pretty much on a constant basis. I got a kick out of this one because it probably represents a lot of the way we feel. It's a five-frame cartoon. In the first frame we have a website with a talk bubble saying, "Welcome to my website. Here are some ads." And the viewer responds, "I won't see them because I have Ad Blocker." Whereupon the website says, "Well, you have to turn off Ad Blocker in order to continue." And the visitor says, "Goodbye," leaving the website sort of there holding its clipboard, thinking, huh, that didn't quite go the way we hoped.

And the point of course is that, given a choice, people may well just say, you know, if I have to lower my defenses, I mean, we've seen lots of instances where you have malvertising, as it's called, where an ad is actually malicious and can do some damage. So there's an argument to be made for blocking ads, not only speeds up the page, makes it look a lot better, but it even increases its safety. And I've often talked to Leo about how I also run with an ad blocker. I just, you know, it makes for a much better web experience.

MIKAH: Right.

Steve: And sometimes I'm using somebody else's machine, and almost I just sort of assume it's going to be mine, or like mine. And then all this stuff starts jumping around and flashing at me, and I think, what has gone wrong? And then I go, oh, this is what the typical web surfer experience looks like. Probably 99.9% of people don't bother with an ad blocker because it's not built in. And so that's what they get.

Anyway, okay, so I don't think that, as you and I were actually talking about before we hit the record button, that last week's podcast was over before I began receiving helpful tweets from our listeners letting me know that it was no longer true that the Windows Print Spooler service could be stopped, and that applications would then be able to print directly to a system's printers, rather than running through the spoolers. Tweeters were informing me that wasn't true. And they were 100% correct. I dug into this a bit because I'm completely sure, as Leo was when he joined me in this belief last week, that there was a time when you could disable the print spooler, and everything would still work.

It turns out that printer dialogs even still, in their configuration, they have a setting for "Print directly to printer," rather than going through the print spooler. So I tried changing that, and it didn't help. Even on my Windows 7 machine, if the print spooler is not running, game over. Doesn't matter how you configure your printer properties. No.

Okay. So for any machines relative to this PrintNightmare problem that may actually need to be able to print something, disabling what has continued to be a vulnerable print spooler service will not be a useful workaround because you need it to print. One thing I guess you could do is if you were really concerned about this, is just run with it disabled, or run with it stopped because that's as good as, although that means it'll restart when you reboot your machine, but normally have it stopped. Start it when you need to print.

MIKAH: Oh, wow.

Steve: And stop it again, although that's a bit of a pain. However, as we're going to see, that actually may be, depending upon your situation, the only solution.

Okay. So as we know and described last week, two different problems have been discovered in the print spooler. There's a remote code execution problem and a local privilege escalation problem. What Microsoft fixed a month ago - and here we are, by the way, on Patch Tuesday for Windows, the second Tuesday of July. It's not clear yet what we're going to be getting. We typically talk about the next week, since often there's some consequences to people who install the updates on Windows, like things stop working that used to. Oddly enough, printing often stops working.

But in any event, what they fixed a month ago with last month's Patch Tuesday was only one of the two. They fixed the local privilege escalation problem. And as we talked about then, they actually didn't do the fix correctly. They stopped the problem as it was submitted in a proof of concept by a well-meaning security researcher. They stopped it from having the problem, but apparently they didn't look around and realize that it was a symptom of a bigger problem which they did not address. So that was the first fumble a month ago. On the other hand, it's good that they did what little they did because escalation, as we know, of local privilege still is an extremely valuable capability for malware to obtain. So anything that can be done to scale that back is good, although not actually fixing the problem is not good.

Obtaining execution of attacker-provided code on any modern OS is also a big problem. But still, any serious exploitation of the machine often requires, even if an attacker can get their own code running, it requires more system privileges than the typical user is now running with on a daily basis. And that's why in Windows we have the UAE, right, the User Account Controller (UAC) and where the screen darkens, and you have to say yes, I'm authorizing you to do something that requires additional privileges. So with this split token design of Windows, you're able to, you know, they've sort of tried to keep people running with minimal privileges while not inconveniencing people who occasionally need to install a program, for example.

Anyway, the bad news is what Microsoft did failed to resolve the other trouble, which was the remote code execution vulnerability. They did something toward limiting privileges, but not keeping random code from being provided remotely and running.

So as we also explained last week, it turns out that over the years, while Windows was quietly requiring the print spooler service to always be running, it was also adding some fancy new on-the-fly printer driver installation options, and therein lies the problem. And it's actually a problem that Microsoft is now saying, uh, we're not going to be fixing that. But we'll get to that in a second. In a big networked office environment, what happens if your local Windows client, the machine that you're perched in front of, doesn't currently contain a driver for some printer in another on-campus building, for example, to which you've been told to send something.

So wouldn't it be slick, apparently thought Microsoft, if Windows could see that it's missing a needed driver for that printer, go find it somewhere, typically from the printer server that it's trying to access, have it installed like autonomously into your local machine for you to then print through that now-present driver to a remote printer that needs you to have that driver in order for you to print to it, and have this all happen in the background. In this case, in answer to our often-posted rhetorical question, "What could possibly go wrong?," we learn that bad guys have figured out how to trick Windows into downloading their malware by disguising it as a printer driver and saying to Windows the equivalent of, oh, no, we need this printer driver now.

Microsoft describes this capability which was added, like way back in Windows 2000, as Point and Print. And so in their description of Point and Print, they say: "Point and Print is a term that refers to the capability of allowing a user on a Windows 2000 and later client" - still here today with us - "to create a connection to a remote printer without providing disks or other installation media." Of course this sounds like this was written back in Windows 2000 when anyone actually had disks or installation media. Like, oh, it's in the box. Oh, I have a box?

So anyway, they said: "All necessary files and configuration information are automatically downloaded from the print server to the client." Isn't that handy. They said: "Point and Print technology provides two methods by which you can specify files that should be sent from the printer server to the client machine. Files can be associated with a printer driver. These files are associated with every print queue that uses the driver. Or files can be associated with individual print queues." And then they go on. "For more information, see Supporting Point and Print During Printer Installations documentation," and so forth.

Okay. So it's not the Point and Print service that's directly being exploited with malicious intent, but rather the operating system's supported underpinnings, that is to say, the API calls which Microsoft created in order to build this Point and Print service. So the CERT Coordination Center's vulnerability note, which was VU#383432, is titled: "Microsoft Windows Print Spooler allows for RCE" - as we know, that's remote code execution - "via," and then they have the name of the API, "AddPrinterDriverEx," E-X.

Okay. And just for our listeners who don't know, Microsoft does this all the time. The Ex is short for "extended" because there is also an AddPrinterDriver. But after a few years, they realized, oh, we need to add some more bells and whistles to this thing. It doesn't do everything we need. So they extend the API, which adds a bunch more parameters typically, and so you always have the original one, and then you have the extended one. So anything ending with Ex is like Rev. 2 or v2.0 of this particular API call.

MIKAH: Does the Ex just become the junk drawer from that point on? Does anything new go into it?

Steve: Well, the good news is we've never seen ExEx. That would be bad.

MIKAH: Okay, yeah, that was my question.

Steve: Typically they say, okay, we're not doing that. We'll come up with a different name.

MIKAH: Oh, gotcha.

Steve: Instead of the extended API. So the CERT Coordination Center is explaining, they said: "The Microsoft Windows Print Spooler service fails to restrict access" - okay, I love this, because they're not Microsoft. Microsoft pussyfoots around this. CERT says: "The Microsoft Windows Print Spooler service fails to restrict access to functionality that allows users to add printers and related drivers, which can allow a remote authenticated attacker" - but that doesn't mean like authenticated with high privilege. "Authentication" just means, okay, yeah, we see you - "attacker to execute arbitrary code with system privileges on a vulnerable system."

And now remember the term "vulnerable system" because Microsoft themselves are going to be using that in a minute here.

MIKAH: Good. I'm wondering what that means in this case. Is it every system?

Steve: Yeah.

MIKAH: Yeah, okay, good that we're figuring that out.

Steve: So they said: "The RpcAddPrinterDriverEx function" - the enhanced one - "is used to install a printer driver on a system. One of the parameters to this function is the DRIVER_CONTAINER object, which contains information about which driver is to be used by the added printer. The other argument, which is dwFileCopyFlags, specifies how replacement printer driver files are to be copied." And actually, because I was curious when I was digging around in this, that's the additional parameter that was added in the Ex version of the API. They thought, oh, we should, in order to do this on-the-fly installation of drivers, we need to somehow specify whether to copy, to replace, what to do as part of this. So this dwFileCopyFlags is what is present in the Ex version of the API.

Anyway, they said: "An attacker can take advantage of the fact that any authenticated user can call" - again, any authenticated user - "can call RpcAddPrinterDriverEx and specify a driver file that lives on a remote server." And that's, by the way, any remote server, like one in Russia. "This results in the Print Spooler service, spoolsv.exe, executing code in an arbitrary DLL file with system privileges."

MIKAH: Oh, my god.

Steve: What could possibly go wrong? Oh. So they said: "Note that while original exploit code relied on the RpcAddPrinterDriverEx to achieve code execution, an updated version of the exploit uses RpcAsyncAddPrinterDriver to achieve the same goal. Both of these functions achieve their functionality using AddPrinterDriverEx." Then they finish: "While Microsoft has released an update for CVE-2021-1675" - that was the fix in last month's Patch Tuesday. They say: "It is important to realize that this update does NOT" - all caps, their emphasis - "protect against public exploits that may refer to PrintNightmare or CVE-2021-1675." In other words, they didn't actually fix the problem that they said they were fixing in 1675.

And then finally they finish with: "On July 1st, Microsoft released a bulletin for CVE-2021-34527" - okay, that's Part 2 -"and its Patch Tuesday, July 6th" - which was last week while we were doing the podcast. "This bulletin," they wrote, "states that CVE-2021-34527 is similar but distinct from the vulnerability that is assigned that 1675, which addresses a different vulnerability in the RpcAddPrinterDriverEx API." They said: "The attack vector is different, as well. 1675 was addressed by the June 2021 security update." Except not fully, as we learned last week.

Okay. So this is where we were last Tuesday when, near the middle of last week's podcast, the patch for the second distinct vulnerability in Windows printing became available. And for those who haven't caught up with or maybe skipped last week, remember that this is all, like, important because these attacks are being exploited in the wild. The cat got out of the bag, the horses have left the barn, the ship has sailed and, in this case, sunk. This is not good.

So this is why during the podcast this was an out-of-band or out-of-cycle emergency patch that Microsoft released. And so we discussed this in the podcast because it happened. And in fact during I think it was the second commercial break I checked Windows to see whether this thing had become available, and it was. So we announced during last week's podcast that, okay, yay, we have the emergency, come-to-the-rescue, out-of-band, you know, Microsoft is here for you. The next day came the news that it didn't work.

So it was on last Wednesday the tech press was bubbling over the news that, okay, having fumbled this already once, but with a different problem where they, like, fixed the demo of the problem but not the problem itself, the tech press said, believe it or not, this emergency thing didn't work. The Hacker News titled their coverage "Microsoft's Emergency Patch Fails to Fully Fix PrintNightmare RCE Vulnerability." In Ars Technica, Dan Goodin's coverage was titled: "Microsoft's emergency patch fails to fix critical PrintNightmare vulnerability." And then Dan added: "Game-over code-execution attacks are still possible, even after this fix is installed." And finally Lawrence Abrams wrote in his BleepingComputer: "Microsoft's incomplete PrintNightmare patch fails to fix the vulnerability."

Okay. And now what is hard to believe is it turns out this was on purpose. Okay. So, okay.

MIKAH: What? What?

Steve: What happened? Yeah, I know. Microsoft's position is that the main concern is that non-administrative users had the ability to load their own printer drivers; right? That's what I read in their own description of this, that everybody could do it. Kids, line up. Okay. So they're saying that's the big problem, and that this is what they have changed. Microsoft wrote: "After applying the updates, users who are not administrators can only install signed print drivers to a print server. Administrator credentials will be required to install unsigned printer drivers on a printer server henceforth, going forward."

MIKAH: Can I pause here for a second? I just want to confirm what it is that you are reporting here. So you're saying that in the original way of things, anyone who had the Windows machine that they got in front of, whether you had administrator credentials or not, could use this to work with a printer that they found on the network somewhere. And there was no, like, sort of a gateway, that you had to be signed versus not be signed. So any printer server with drivers on it, whether it was signed or not, could be accessed by any person.

Steve: Uh-huh. Air quotes, good. Right.

MIKAH: Okay. So they didn't have something in place to sort of scan these printer drivers to make sure that they weren't bad things.

Steve: Okay. So yes, sort of.

MIKAH: Okay.

Steve: What it turned out to be is that there was a test for authentication.

MIKAH: Okay.

Steve: That is, there was a test in that API where the authentication function, I think it was called ValidateAdminUser, I referred to it last week, where that function would be called to make sure that the user was an administrator who was doing this. But it turns out, and this was what was patched, there was a flaw in the code. And the bad guys found the flaw in the code that allowed that test to be bypassed.

MIKAH: Oh, okay. So they didn't even have to face the test, essentially.

Steve: Exactly.

MIKAH: Okay.

Steve: They just, like, didn't take the test. Okay. So the CERT Coordination Center's vulnerability analyst, who we also referred to last week, he's still on the job, Will Dormann. He cautioned that the patch, and this is his quote, "only appears to address the remote code execution, the RCE, via SMB and RPC variants of the PrintNightmare, and not the local privilege escalation variant."

Okay. So that would allow attackers to abuse the local privilege escalation to gain system privileges on vulnerable systems. Further testing, which the security community then jumped on, has revealed that exploits targeting the flaw could bypass the remediations entirely to gain both local privilege escalation and remote code execution, even though Microsoft led us to believe otherwise.

Okay. In response to that, Microsoft updated their online report, and they're now saying, yeah, uh-huh, that's right. That's on purpose. That's the way it's supposed to work. Okay. What they actually said was, and I'm quoting them, they are clearly stating that this behavior is deliberate. They go so far as to write in their bulletin the following, which I put in the show notes, took directly from the bulletin. And they're referring to two different values in the registry. They wrote: "Having NoWarningNoElevationOnInstall set to one makes your system vulnerable by design."

MIKAH: Vulnerable by design.

Steve: And as you know, it's been staring us right in the face this entire time. We've needed a new slogan to go with the new Windows.

MIKAH: Vulnerable by design.

Steve: And now we have it, yes. Windows 11 - Vulnerable by Design.

MIKAH: I love it. I hate it, but I love it.

Steve: Thank you, Microsoft. So Microsoft's updated bulletin explains it precisely. They said: "In addition to installing the updates, in order to secure your system, you must confirm" - you, it's up to you, you must confirm, apparently they're not going to do it for you - "you must confirm that the following registry settings are set to zero or are not defined." And they said: "Note: These registry keys do not exist by default, and therefore are already at the secure setting. Also," they say, "that your Group Policy settings are correct. See the FAQ."

Okay. So the registry key is HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\ - because, after all, this started back then - Printers\PointAndPrint. And under that key there can be two values. One is named NoWarningNoElevationOnInstall, and there can be another value named UpdatePromptSettings. They're normally not there. But point and click, if you want the point and clickness, then you turn these things on. So you would have those named values set to one in order to turn on point and click.

MIKAH: Point and print. So point and print is not on.

Steve: I'm sorry, point and print, thank you, yes, point and print. Right. It's not there automatically, but it's this feature that they think is great. So what happens is, because there was no known downside, lots of things turn it on. It's on in enterprise settings.

Various additions to Windows that you might install would go, oh, we want point and print turned on. So they would add those keys and set their values, their DWORD values to one. So it can happen.

Last week we showed a flowchart that at the time looked complex. Now it's gotten about twice as complicated with all of the decision trees and tangled arrows that you follow around in circles as you answer questions about what's on, and what's off, and is this key present? And if so, what's it set to? And follow the arrow to figure out. You end up at one green box, which is not exploitable, and then you've got two red boxes, which are locally exploitable and locally and remotely exploitable. And you have to be very careful about which decisions you make in this flowchart, or you end up at red rather than - oh, wait, I'm sorry, there is a green one at the bottom, and I had to scroll down further because the chart is so long, that is not remotely exploitable.

So really today where we are still is the only way to really be safe is to disable your print spooler, or go to that key and make sure it's got no named values. Go to HKEY_LOCAL_MACHINE \SOFTWARE\Policies\Microsoft\Windows NT\Printers\PointAndPrint, and kill off the NoWarningNoElevationOnInstall and the UpdatePromptSettings. Delete them. Everything will still work except whatever PointAndPrint does. I mean, you're turning that off; right? But if you don't - oh, and this is where they say right below that having NoWarningNoElevationOnInstall set to one makes your system vulnerable by design.

MIKAH: God.

Steve: Ah, yes. So it appears that Windows 7, 8, 8.1, Server 2008, which you know is the server version of 7, and 2012, which is the server version of Windows 8, will always be vulnerable by design, even after being patched, because those keys were not supported until later. But Windows Server 2016 and 2019 and Windows 10 and 11 require the PointAndPrint to be configured. So after applying the patches, the later versions of Windows will no longer be vulnerable as long as those two registry keys do not exist or are set to zero. In other words, apply the out-of-band patch and be sure that those keys are missing.

And you know, stepping back from this, it must be that, at least in the short term, because this was after all an emergency patch, in the short term Microsoft was unable to robustly fix this trouble with a simple patch without also disabling some functionality that some of their users rely upon. Right? Like whatever the PointAndPrint is, apparently it's important to some people. So they couldn't just kill it. And PointAndPrint probably requires that a non-admin regular standard user be able to point and print something, that requires this to happen. So as they said, if this is enabled, this is vulnerability by design for this feature. It does make me think that now that this has come to light, maybe they're going to come up with some way of making you point and print and authenticate.

MIKAH: Yeah.

Steve: You know, add another step to this.

MIKAH: Yeah.

Steve: In order to, you know, making it somewhat less easy, but make it not vulnerable by design. The problem is that malware can do this using the API which would otherwise, you know, you'd be pointing to and printing from. They could do it using the API underpinnings behind the scenes. And thus the problem.

MIKAH: See, and this is the thing, is printers just are terrible things in general, and anything that makes them a little bit easier to use should be a feature that we would be able to celebrate. So basically Microsoft is saying this convenience will cost you. So basically don't use it if you want to keep things safe. But that means that you have to do the whole drive dance, as it were, to get any of these printers working. And if you think about it on a college campus, for example, if some student wants to print something from the local printer, and then they've got to figure out what drivers they're supposed to use, take it into the IT, that's just - this feature seems like it would be a good thing. But instead of fixing it, they're just saying no, just don't use it if you want to be secure.

Steve: Well, and the other thing, too, is reading between the lines, it does look like signed drivers are still okay. And so what must be the case is that the world already has so many unsigned drivers that it's not possible to retroactively enforce this.

MIKAH: Gotcha.

Steve: If you were to say, sorry, point and print only if signed, well, you might as well not bother because drivers just haven't been. And so Microsoft has managed to enforce driver signing for the kernel, for like mainstream device drivers. But printer drivers have been allowed to be an exception. They don't have to be signed because, like, again, it's too late to add that requirement after the fact. They were able to do it for Windows moving forward, but not enforce that requirement on printer drivers just because, again, they're just, you know, they're already out there unsigned, so what are you going to do? And so the bad guys are taking advantage of that fact by having malware which is also unsigned, pretending to be a printer driver.

And so for the time being, I guess my point is that hopefully they're going to address this by adding the User Account Control dark screen scary dialog to the point and print, where it's going to be point and print, and are you really sure? And that takes it away from the automatic background operation that the bad guys would be able to deploy as it is right now. As they said, vulnerable by design.

MIKAH: Good golly. What a phrase, "vulnerable by design."

Steve: And well named; you know? PrintNightmare. The guy who named it didn't even realize that we were going to have a hard time waking up from this nightmare.

MIKAH: Yeah, it keeps going and going and going, it seems. What's next, Steve?

Steve: So bad as it was, it turns out that, now that the dust has settled a little bit from the Kaseya crazy, like largest in history, attack, it could have been a lot worse. And I guess the right way to say this is it turns out it wasn't as bad as we thought it was going to be. Despite the fact that somewhere on the order of 1,500 end-user organizations were attacked with the REvil ransomware, unfortunately, using Kaseya's VSA servers as their way into or onto those networks, it turns out that something quite significant was absent from the Kaseya VSA server base attacks. We've become quite used to the individual nightmare stories that follow these attacks. It turns out that something that the attacks did not do turned out to be more significant than people appreciated.

Okay. So as we know, when ransomware gangs conduct the typical attack, they breach a network to actually get into a victim's system, onto their network. They set up shop there, take a good long look around, figure out where they are, what's valuable, like where the goodies are, establish some mechanism for persistence, and thoroughly survey the territory. Then, as a means of increasing their extortion leverage, they'll exfiltrate sometimes bracing amounts of internal private and probably confidential data. You know, we've heard numbers like 700TB of data, just phenomenal amounts of data. And of course they use that to increase the pressure on their victim to pay up.

Once they've got the data exfiltrated, they'll carefully wipe and completely eradicate any and all backups that they've been able to locate. And then, as their last act, they trigger the encryption of all of the machines that they've had access to or found access to everywhere. But that's not what happened during the Kaseya attacks. As I noted above, the REvil affiliate simply used their Kaseya VSA access to download and run the Sodinokibi encrypting ransomware. No network penetration was performed. As I mentioned last week, there were indications even then that no internal data had been exfiltrated. Now we know that to be true for certain. And no backups were located and wiped.

Although it's not nothing to have encrypted all of an organization's computers, it turns out that it actually makes a huge difference when the ransom price of a per-client decryptor is $5 million, and the client discovers to their tremendous relief that all of their backups are still there, in place and usable. So the result of what turned out to be sort of half-baked ransomware, even though it was a blitz, it affected 1,500-plus organizations, it turns out that in this case ransoms are often not being paid. Instead, encrypted systems are being restored from backups rather than being decrypted.

Without the data exfiltration and no way to restore except by paying ransom and obtaining a decryption key, that REvil affiliate lost a huge amount of leverage over their victim organizations. Multiple organizations and an MSP told Bleeping Computer that none of their backups were affected, and that they chose to restore from backups rather than paying a ransom. Yeah.

MIKAH: Yeah, who wouldn't?

Steve: Yeah. Do the math. Bill Siegel, who's the CEO of Coveware, which is a leading ransomware negotiation firm, told Bleeping Computer that this is what they also had been seeing, and not a single one of their clients has had to pay ransom. He said: "Impacted MSPs are going to be stretched for a while as they restore their clients, but so far none of the clients we have triaged have needed to pay a ransom." He said: "I'm sure there are some victims out there who will need to, but this could have been a lot worse."

So it's great news that a week or two out, we're not talking about 1,500 individual and extremely expensive disasters as a result of the single Kaseya VSA server mess. Questions have been raised about how the hackers knew of the vulnerability. Did the news that it was soon to be patched somehow leak? Did they know that they only had a brief window of exploitability, so they may have been in a rush? If they were the typical REvil affiliate, they surely had to know that not entering individual networks to survey, exfiltrate, and wipe backups would dramatically reduce their leverage.

But perhaps they hoped to make it up in the numbers because this was a huge, huge attack. Perhaps they assumed that they'd score enough ransoms to make up for those who would choose to restore from backups. Or they hoped that many smaller organizations still weren't doing backups. As Bill Siegel said, he did expect that there would be some ransoms paid. So let's hope that this does remain the biggest ever ransomware attack in terms of numbers of compromised organizations, and that we don't see that being exceeded, and that it actually turns out to be one of the smaller attacks on balance when you take a look at the actual number of ransoms that had to be paid.

MIKAH: Now, what about - so sure, they were able to, or they had the backups. But doesn't it also mean that that data is still stolen and out there? So there was still a data breach. What does one do about that? Did they just notify all their clients, hey, all of your personal information is out there on the web now?

Steve: No. That's actually what was so cool is that the servers that were vulnerable had an authentication problem that allowed the attacker to use the server only to do an auto update to their clients' computers. That auto update was malicious, and it did the encryption.

MIKAH: Okay.

Steve: But it did nothing else.

MIKAH: Okay, gotcha.

Steve: All they were able to do was to push the ransomware out to these client systems. It would then encrypt everything it could, but it performed no exfiltration.

MIKAH: Gotcha.

Steve: So there was no data leak as a consequence.

MIKAH: Okay. I gotcha now. Thank you.

Steve: Yes. So made it way better. Okay. So anybody interested in ransomware, and you guys are going to want to bring this page up while I'm talking about it, needs to go to Ransomwhere, W-H-E dot R-E. So it's R-A-N-S-O-M-W-H-E dot R-E. It is really fun. It's a guy named Jack Cable who just graduated, he's Class of '21 from Stanford, and he's working now with a security group. He created this site. Again, Ransomwhere with a dot RE at the end.

So far they're tracking individual transactions, ransomware payment transactions, totaling $60,274,058.06. From the site we learn that NetWalker holds the all-time record, and it's followed in decreasing order, not surprisingly, this is exactly the order we would expect, by REvil with Sodinokibi, Ryuk, DarkSide, RagnarLocker, Egregor, Conti, the BitPaymer-DoppelPaymer, that's the same thing named two ways, then SynAck, then Qlocker. And he shows also a cool ransom payment chart showing the ransom, the bitcoin address to which it was paid, the date that the transaction occurred, and the amount in bitcoins. Oh, and also the hash for the transaction. He also has a section showing the latest attack reports as they've been coming in, and provides a form for sending him news of new reports, along with all those details.

The top three items on his site's FAQ are, one, asking the question, why track ransom payments? To which he answers: "Transparency is crucially needed in assessing the spread of ransomware and the efficacy of mitigations. Fortunately, due to the transparent nature of Bitcoin, it's easy to track payments with knowledge of receipt addresses. By crowdsourcing ransomware payment addresses, we hope to provide an open resource for the security community and the public."

So then the next question of the three I chose at the top, how complete is the data? And he says: "As Ransomwhere is new, we're still working on building out our dataset. Reports have placed total ransomware revenue in 2020 at up to $350 million." And so the point they're making is they've locked down the details for 60 million, so there's a lot that is still missing. They're hoping to collect it, and they're looking for input.

And to that end, the third question, can't someone fake a report? To which he replies: "While it's important to verify with complete certainty that a report is accurate, we aim to utilize the wisdom of the crowds to prevent abuse. All reports are required to include a screenshot of the ransomware payment demand and will be reviewed before being displayed. Addresses with more than one report to substantiate them from different sources will be given priority, and all elements of all reports will be publicly available. We will remove reports if we believe they are untruthful." So it's cool. It's an open clearinghouse for reports to which anybody can submit who has hopefully truthful knowledge of a report. They'll do the vetting that they can before adding it to their knowledge base. But I think another cool resource, Ransomwhere, with a dot RE on the end.

There was also news, it was actually McAfee-sourced, some original detailed reverse engineering of a Microsoft Office-based new malware protection bypass. And I thought it was interesting to talk about just because we've sort of talked about problems with Office in the past, and Word macros, and oh, those dumb email people clicking on links that they shouldn't. Okay. So let's take a look inside this because, thanks to McAfee's report, we have some insight into how this one particular new attack that has never been seen before is operating. So I wanted to walk everyone sort of through the top-level design and operation to get some sense for what's going on.

"Zloader," as it's known, is a well-known banking trojan which, if it gets into your system, it will steal credentials and other private information from the users that it's targeting at the financial institutions that it knows about. It'll do things like watch your keystrokes and recognize when you're entering information which it can see connects to an institution that it's aware of. So it's sort of, you know, think of it as a vertical market banking trojan, as they tend to be.

So this new attack uses Office's Word and Excel in conjunction, essentially working cleverly in concert, to disable Office's macro warnings, and to then download and enable this Zloader banking malware to get into a user's computer without triggering any security warnings, and flagging it as what it is, you know, something you don't want in your machine. The initial attack vector is a standard email inbox-based phishing message containing a Word document attachment which in itself contains no malicious code. So something scanning all incoming email will think, okay, huh, there's a Word attachment. But it doesn't look like there's anything bad in there, so typically would not trigger any email gateway scanner or client-side antivirus software which would be attempting to block something that it recognized as malicious.

MIKAH: Hmm.

Steve: Uh-huh.

MIKAH: That's clever.

Steve: Since Office automatically disables macros, the attacker's email attempts to trick the naive email recipient into enabling macros, with a message appearing inside the Word document. The Word document explains: "This document was created by a previous version of Microsoft Office Word. To view or edit this document, please click the 'Enable editing' button on the top bar, and then click 'Enable content.'"

MIKAH: So, okay. So then there's part of the problem is that they've called it "enable content" instead of "enable macros" specifically. So just by...

Steve: Yeah, or if it said "enable probably malicious malware."

MIKAH: Yeah, why don't they rename it that?

Steve: You know, that would stop a few people. They'd go, well, no, okay, wait a minute. Is that what I want to do this afternoon? Right. Exactly. It's called "enable content." It's generic, and it doesn't explain the consequences.

MIKAH: Right.

Steve: And since people transacting, whose lives are about Word document attachments, they will likely have encountered similar legitimate Word version conversion messages in the past. I've seen them. The document's legitimate-appearing instructions, like when this actually happens, it's like, oh, this document was from Word 2003. We'll have to convert it. And you go, yeah, okay. I've had that. I've done that. And I didn't get myself infected because it was real. And so the point is they're making this look like something that the person who does this sort of thing routinely has seen before. And so it's like, oh, yeah, okay, fine, yeah, got to do that. Click here. Click there. Fine.

Okay. So now the fun begins. Visual Basic for Applications; right? VBA. This is code which is embedded in the Word document which is able to use another longstanding feature of Windows known as DDE (Dynamic Data Exchange). It's able to read a specially crafted .XLS spreadsheet from a remotely located foreign domain, which it uses DDE to load this into an Excel cell in order to create and then run an Excel macro. So this is sort of - this is a sneaky way of bootstrapping benign-looking code which then uses Dynamic Data Exchange to download some text from a foreign location, which it then sticks into an Excel cell, turning it into an Excel macro, which is not benign. But everything that happened up until then was.

That macro, in turn, populates an additional cell in the same XLS document which was created by VBA with another VBA macro, which disables Office's defenses. The Word document, it turns out, is able to indirectly set the policy in the registry titled "Disable Excel Macro Warning." So it says, yeah, let's disable Excel's macro warning, which it does, after which it invokes the malicious macro function in the Excel file. That causes Excel to download the Zloader payload, which it's able to execute on the fly from RAM, using the rundll32.exe, which is a common way of running code in a Windows system.

So what we have is a back-and-forth interaction between two very powerful products, Word and Excel, which have both grown into incredibly complex application-hosting containers. And thanks to a bit of benign-appearing social engineering, an unwitting user, doing nothing more than they have probably done many times before, allows that foreign code to be executed and downloaded into their machine and run. So we find ourselves once again back wondering whose fault it is. There's no doubt that the power of Word and Excel have enabled marvelous capabilities and the construction of valuable applications, all on their own. People write whole apps in Excel. It's like, oh, yeah, here's an app I wrote. It's an Excel spreadsheet. It's like, what? Yeah. Because it's got a fully complete language that it's able to operate.

And it's also true that users of these programs which have been around for many years and which have gone through many changes and versions, that is, like Word, old versions of Word, will often receive actual requests for actions and conversions based upon Word version document mismatches. So it seems to me that the problem is that email, I mean, this is still where the crap comes into a person's machine. Email is still, in 2021, not being regarded by Microsoft as a sufficiently dangerous vector. Any fair reading of the evidence would lead to the conclusion that this appears to be a huge blind spot for Microsoft. Malicious macros hosted by Outlook email predate this podcast's first episode nearly 16 years ago.

MIKAH: Wow.

Steve: We've always been talking about them. Like how long does it take Microsoft to learn this lesson? And at this rate it doesn't appear this podcast is going to outlive them. They were here before, and they will be here afterwards. A huge amount of effort has been made to successfully sandbox our web browsers because they represent an obvious source of externally provided, potentially malicious content. Why is email regarded any differently? By definition it's externally provided. And if it's allowed to have executable Office document content attached, which includes Visual Basic for Applications code, the profile of its attack surface is no different from a web browser's, which is running JavaScript from lord knows where, offered to you from a server that you have no control over.

This is one of those maddening things where we keep hearing about the same problem over and over, year after year, yet nothing gets done about it. So what do we inevitably wind up doing? We blame the user for being duped when they behave in a perfectly natural way. It's not right.

MIKAH: I couldn't agree more. Especially with that last part. Because you have this situation where you're going, oh, people who have a little bit more knowledge about this stuff, who maybe lack the empathy that you are actually bringing to the table here in that statement, who go, you really clicked on that? You really got duped by that?

Steve: You dummy.

MIKAH: You were really tricked by that? Exactly. And something like that, you even said it, and I've, yeah, I've gotten a prompt that says, oh, this is a .doc file, would you like to make it into a .docx file, which is the more modern, blah blah blah blah. You click okay. And again, I'm still upset to hear that they're calling that feature "enable content" instead of breaking it out into different things including "enable macros." When you don't know - because I could see, you know, they get a coaching session from IT, and in it, it says never turn on macros.

Steve: Never enable macros, yes.

MIKAH: Right. But then to take that and go, oh, but this is saying "enable content," and I've gotten this prompt before, to just immediately go...

Steve: Yeah, and I want the content. That's the whole point; right?

MIKAH: Yeah, exactly. I'm trying to get to this email. This is what I'm supposed to have. So, yeah, I agree with you. It's one of those things where you look at how this has been an issue for so long, and you're going, how was this system enabled? Like all the stuff, you're talking about the VBE, or excuse me, the VBA, and the Dynamic Data Exchange. I didn't even know that was possible in the first place. And these people are so, so, so, so clever at coming up with these new social engineering methods. And it's, wow, it's wild.

Steve: Yeah, it was designed to be powerful. Unfortunately, it is.

MIKAH: Yeah, exactly. We made it this way, and now it's working as expected.

Steve: That's right. We made it so powerful you can point and click, and so can the Russians. Wonderful. Okay. So turns out, speaking of Russians, what do you do if all your victims speak English, but you only speak Russian, yet you need to interact with them to negotiate a ransom settlement? Turns out that the role of bilingual ransomware negotiators is now in high demand. Again, the inevitable sometimes happens. We've talked - I guess, wait, does the inevitable always happen? That's why it's inevitable, yeah.

Anyway, we've talked about the evolution of the ransomware ecosystem and how there are now penetration providers - penetration providers - who specialize only in breaking in, and that's as far as they go. Those break-in penetrations are purchased by ransomware as a service affiliates, who are working on behalf of the ransomware service providers, who then move into the victim networks to inventory, exfiltrate, wipe backups, and eventually encrypt a network. And on the victim side we have the ransomware attack insurers and the emergence of third-party ransom escrow providers.

MIKAH: Wow.

Steve: So, you know, at all levels we're getting specialization. So to this milieu we now add negotiators who are bilingual interpreters interposed between the attackers and their victims, and maybe their victims' agents. A recently published study of ransomware as a service trends revealed that the additional go-it-alone ransomware operations have virtually disappeared in favor of multi-tiered organizations. Literally, I mean, that's what you would call it; right? It's organized. It's an organization. And most recently this has led to a high demand for individuals to take over the negotiation part of the attack chain.

There's a group, KELA, K-E-L-A. They published a study that explained a typical ransomware attack comprises four stages: malware code acquisition, spread and infection of targets, the extraction of data and/or maintaining persistence on impacted systems, and then monetization. Right? Getting paid. There are actors, they said, in each area, and recently demand has been increasing for extraction and monetization specialists in the ransomware supply chain.

The emergence of so-called negotiators in the monetization area in particular, they said, is now a trend in the RaaS, ransomware as a service space. KELA researchers say that, specifically, more threat actors are appearing that manage the negotiation aspect as well as piling on the pressure though telephone calls, DDoS attacks, and making threats including the leakage of information stolen during a ransomware attack unless a victim pays.

The newly clarified role has emerged due to the need for individuals able to manage conversational English in order to effectively conduct the many aspects of post-attack negotiation. A spokesman for KELA said: "This part of the attack also seems to be an outsourced activity, at least for some affiliates and/or developers. The ransomware ecosystem, therefore, more and more resembles a corporation with diversified roles inside the company, and multiple outsourced activities."

So we talked about the so-called Initial Access Brokers (IABs), as they're now called. They're also seeing increased demand. After observing dark web and forum activity over a year, the researchers said that privileged access to compromised networks has surged in price. Some listings are now 25 to 115% more than had previously been recorded, especially when domain admin-level access has been achieved and is being offered on the dark web. These "intrusion specialists" are now receiving a piece of the action between 10 and 30% of the ransom payment.

So, yup, we're seeing stratification, specialization, corporation, organization. We're now having people needing to be bilingual in order to cross the language barrier that exists between those who attack and those who are attacked. And since getting into a network is like the first step of any successful attack, affiliates are now willing to cut the intrusion specialists, the so-called "initial access brokers," in for a piece of the action, 10 to 30%.

MIKAH: So I assume at one point there had to be concern about all of these bits of the market cropping up, that that lent a certain level of legitimacy to - it's like, what is it, we don't negotiate with terrorists. But at this point it's probably too far, it's too gone - it's too far gone now; right? I mean, what do you do once this market does start cropping up, and you have all of these different fields? You can't really say no, we're not going to have translators; and, no, you can't put that on your rsum. Because I did see there was some warning for some insurance companies that they weren't supposed to pay out for ransomware stuff. And, I mean, what do you do?

Steve: Right, right. I mean, so what I think we're going to see, and it's sort of surprising we haven't yet, there will be some ransomware attackers who do not honor the code which the big guys have been careful to maintain. We're going to be talking about REvil's Clever Crypto here in a few minutes. And part of the design, as we'll see, provides the original REvil creators the ability to decrypt anything that any of their affiliates are able to encrypt. So that allows them to maintain ultimate control so that, if an affiliate disappears, they're able to step in in their place.

Or in the worst case, and this has happened, there was an interview that we covered last week where someone, one of the REvil guys, was asked, have you ever encountered a problem with decrypting somebody's data once they paid ransom? And they said yes, when an antivirus gets in and mucks with some important files, we may not be able to decrypt it. We refund their ransom.

MIKAH: Interesting. Interesting.

Steve: So there is at the moment some honor among the bigger guys because they recognize when they say "Pay us $400,000 in bitcoin and, A, we will give you the ability to decrypt your systems; and we will delete all of your files which we have exfiltrated." Well, it is crucial that that be honored, or they realize nobody is, you know, they'll suffer the pain of the loss of their data if they don't think that the $400,000 they pay is actually going to get them restored and protect the data that's been stolen. The problem is, all it's going to take is some greedy second-tier ransomware group to start dishonoring what the real ransomware guys have been deliberating honoring for this whole thing to start getting some holes shot in it, exactly as you suggest.

MIKAH: Interesting, okay.

Steve: And I'll bet we're going to see that.

MIKAH: Yeah, probably very soon, as other people start to try to move into this space, and they're not...

Steve: Oh, yes, exactly. Who would not want, well, you know, good guys want to. I'm not doing it. I could write that software; but no thank you. I've got better things to do. But the more these attacks are advertised, the more the size of the ransoms become known, the more script kiddie-ish kind of guys are going to be getting in, saying, hey, like asking on the dark web, will somebody write me some ransomware because I've got some people I want to attack. It's like, oh, great, you know, that's not going to go well.

MIKAH: Oh, dear. Yeah, that's scary, too. Now suddenly you've got people paying for a ransomware program that they can install on some company's machines.

Steve: Right, which they didn't have the ability to write themselves. So, yeah.

MIKAH: Wow. Wild.

Steve: Okay, a couple pieces of miscellany. I titled this one "You will use the new Start Menu, and you will like it." Last Thursday, Microsoft released Windows 11 build 22000.65 to Insiders who are on their dev channel. And believe it or not, support for the manual registry value that could be added to cause earlier pre-release builds of Windows 11 to revert to a Windows 10-style left-aligned Start menu had been removed. It's like, really, Microsoft? Really? Come on.

MIKAH: There's no reason to do that.

Steve: Really? The code is there. You have it in there. You're going to say no. At this time it is no longer possible to ask Windows 11 to continue using the Win10-style Start Menu that many have grown used to and wished to continue using. All I can say is hopefully Microsoft will choose to surface an official option for what they might call the "Classic Win10 Start Menu" somewhere in the Windows 11 settings screen personalization stuff because that's just really, you know...

MIKAH: That's so silly to not give that option.

Steve: It's just childish. We're going to force you to use what we think is better.

I also don't have anything earthshaking to report since last week on the SpinRite front. I'm at work rewriting SpinRite's core, converting from its historical track-at-a-time, you know, track-based design to its new linear sector stream architecture. Once that's finished, I'll first put everything through its paces myself. Then I'll turn the GRC newsgroup gang loose with the code, and that will become the pre-release of SpinRite 6.1. That is to say, once they stop being able to make it misbehave in any way. That'll be the pre-release of SpinRite 6.1, which we'll call the beta.

People have been writing and asking for the 6.1 beta already. So I just wanted to clarify that nothing yet exists for anyone to use. I can't even use it yet. I'm still working on just finishing the last chunk of it. As soon as it does exist, it will be made available in beta form for any existing 6.0 user to download. So definitely stay tuned. It's what I'm spending all my time on, and we're getting close.

MIKAH: Awesome.

Steve: Okay. So this is going to be a little bit of a - we call them the "propeller head" episode. But it's good stuff.

I saw somewhere, the way this began, that Kaspersky Labs wrote: "REvil uses the Salsa20 symmetric stream algorithm for encrypting the content of files and the keys for it with an elliptic curve asymmetric algorithm. Decryption of files affected by this malware is impossible without the cybercriminals' keys due to the secure cryptographic scheme and implementation used in the malware."

And then, as I noted last week, we also learned that it was possible to obtain decryption keys for files of a given file extension, or for an entire single enterprise, or a single system, or for all enterprises that were victims of an attack campaign. So this was like a surprising amount of granularity of what could be done with this cryptographic system. So I wanted to explore and share the design of REvil's Clever Crypto.

First off, I've noted that most of the tech press is unaware or technically inaccurate about their use of the term REvil versus Sodinokibi. I try to use them correctly where I can, but I'm often needing to quote their misuse. So just for the record, REvil is the gang, and Sodinokibi is their encrypting malware. Even though I titled this episode REvil's Clever Crypto since it rolls off the tongue more easily, it's the clever crypto employed by the REvil gang in the design of their Sodinokibi malware. Just to get it correct.

Okay. So before I describe the awesome cryptographic design employed by Sodinokibi, I want to quote something from Acronis's broader description of their reverse engineering of Sodinokibi. To any non-Windows coder, this is going to sound mostly like a bunch of mumbo jumbo connected by a few recognizable bits of English.

MIKAH: I'm ready.

Steve: But, yeah, hold on. Luckily you're on a chair and not on Leo's ball because, you know, just might have a stability problem. Okay. I think it's worthwhile to set the stage for understanding what we're facing with the quality of the design of Sodinokibi. So here's what Acronis found.

They wrote: "To encrypt user files, Sodinokibi uses I/O completion ports" - and I'll explain what all these terms are in a second. But I want to just give you a sense for it. "Sodinokibi uses I/O completion ports and creates multiple execution threads to a maximum of twice the number of processor cores present on the machine, and associates these threads to the created I/O completion port. These threads use the GetQueuedCompletionStatus Windows API function to wait for a completion packet to be queued to the I/O completion port before they proceed to the file encryption.

"Once the threads are created and waiting for I/O packets to arrive, Sodinokibi starts enumerating user files on all the local drives and network shares." And I put in my own comment here. For anyone who may have been wondering whether Sodinokibi followed network shares, uh-huh, it does. Anyway, except, they wrote, "CD-ROM and RAMDisk drives. And it begins associating files which are not in the exempted folder, file, or file extension lists to this I/O completion port by calling their user function AddFileToIoCompletionPort and calls the PostQueuedCompletionStatus Windows API to post an I/O packet to the I/O completion port, which will trigger the thread waiting on this I/O completion port to resume and proceed to encrypt files." We're almost done.

They said: "The user function AddFileToIoCompletionPort also generates a unique Salsa20 key for each file that is to be encrypted, and passes this Salsa20 key to the encrypting thread as part of the structure containing other metadata that has to be written to the file as well after encryption using lp" - that's long pointer - "lpOverlapped parameter of PostQueuedCompletionStatus Windows API. During enumeration, it also creates a random note file in all folders that are not in the exempted folder list. Once there are no more files to enumerate, the main thread waits in a loop until the total number of files encrypted and renamed equals the total number of files added to the I/O completion port for encryption."

Last sentence here: "Finally, it sets a flag which indicates that there are no more files to enumerate and posts multiple I/O completion packets. By doing this it makes sure that extra threads waiting for files should resume and break the execution flow to finish immediately." Okay, now...

MIKAH: Simple.

Steve: I've been programming Windows for several decades. And as we all know, I write in assembly language at the Windows API level. What I just read not only makes perfect sense to me, but it is precisely the optimal maximum performance multi-threaded design for working with any large collection of objects under the Windows API. And in fact this is precisely the design I use to manage all of GRC's web server side services where a lot of stuff needs to be going on at the same time.

So what I read, all that gobbledygook, amounts to using a low-level native Windows API to create a queue of file encryption jobs which actually exists in the kernel and will be serviced by a pool of processor core threads. The reason this is so efficient is that jobs are removed from the queue explicitly avoiding expensive thread context switches. Typical queues would wind up by assigning jobs to threads in a round-robin fashion where the longest waiting thread would get assigned the next task. But Windows' completion port design, first of all, it's kernel based, so it involves ring 3 to ring 0 transition delays. And it deliberately is designed to allow the same thread that's executing at the time and is posting its completion status to obtain the next job of work on the queue, and to begin working on it immediately without using its scheduling time slice.

So to put this into context, for their application, it's over designed. It's a small matter. In the case of Sodinokibi, where the individual jobs will be comparatively large and long-running compared to a threat context switch, just the context-switching overhead would be undetectable. But what this showed me was that whoever wrote this piece of code, they really knew their way around the Windows API. They weren't taking shortcuts. It wasn't written in some high-level abstraction. They knew exactly what they were doing and how to do it.

So with that preamble, let's talk about the crypto. When interviewed, a member of the REvil gang expressed their extreme pride in their creation of Sodinokibi. As a researcher commented, this pride was warranted because Sodinokibi is one of the few ransomware families that actually thought about its cryptographic scheme and how to account for all various situations.

Okay. So there are four applications of and types or instances of public/private key pairs used by Sodinokibi, as follows: There's an operator key pair whose public half is hardcoded and built into every Sodinokibi code sample that has been obtained. This value is set by the REvil operators themselves, the REvil gang, and cannot be set or changed by an affiliate. So in other words, there's an operator public/private key pair, the public component, the public half of which is built into an affiliate's code, which the affiliate then goes out and does evil with. The private key is maintained by REvil and is kept secret.

Okay. There's also a campaign key pair whose public half is stored as the PK value, as it's called, in Sodinokibi's per-victim or per-campaign configuration, depending upon the application. This value can be modified by the affiliate. So the campaign key pair, again, the public component is like per-configuration. The affiliate keeps the private key, the private half of the campaign key pair, as their secret, much as the REvil gang keeps the operator, the private operator half of their secret.

Then there's a system-specific key pair, that is, like a machine-specific key pair, that's generated as an individual system is being encrypted. The private half of that key pair is encrypted in parallel, as opposed to in series. The private half of that key pair is encrypted using both the operator's public key pair, or public half of its key, which is built into the code, and the campaign's public key, which is in the campaign's configuration. And lastly, there's a unique per-file key pair which is individually generated...

MIKAH: Oh, my word.

Steve: ...individually generated for each encrypted file, separately. The public half of the per-file key pair is stored in plaintext within the Sodinokibi file header that's appended to the front of the encrypted file content.

Okay. So, now, here's where things get really cool. Bulk file encryption is, as Kaspersky alluded to, performed using Dan Bernstein's excellent and quite fast Salsa20 symmetric cipher. The symmetric cipher, which will be used for both the encryption and maybe, if you pay the ransom, the decryption, is obtained by combining the private half of the per-file key pair, which they didn't say it, but I know it because I did the same thing in SQRL, you actually discard this. You combine the private half of the per-file key pair with the public key - my notes are jumbled here because I know what I meant, but I didn't write it.

So the symmetric key is obtained using ECDH, which is the Elliptic Curve Diffie-Hellman algorithm, the Elliptic Curve Diffie-Hellman key agreement, under Curve25519, which is the same one I used. The idea with Diffie-Hellman is that, just to remind people, is you can have two people distant from each other. They each create a private and public key, that is, a private and public key pair. They each send to the other the public side of their pair. And upon receiving it, they each take the other person's public half and their private half, run it through Elliptic Curve Diffie-Hellman, each of them, and they get the same result. So this is what's so cool is that it's amazing because...

MIKAH: How does that work?

Steve: I know. It's math, baby. And what's so cool is that a man in the middle, somebody eavesdropping, even the NSA, all they see is each of their public keys passing, and they don't ever get, no observer gets the private side. So both public keys tell you nothing. It's only by combining the matching private key with the other person's public key that you're able to get a result. And what's so cool is you both get the same result. So this is the application of that for communication. It turns out you can use it for encryption. And I did that in SQRL. I did exactly the same thing in SQRL. So with Sodinokibi, they take the private half of the per-file key pair with the public half of the system key, combine those using Elliptic Curve Diffie-Hellman to get the symmetric key which will be used to encrypt the file using Salsa20.

Okay. So the key takeaway here - ignore the pun - is that only the private half of the system key is required to decrypt every file for that system because you can reverse this, essentially. Each file to be decrypted provides its public key, which was in its header, along with the private side of the system key. That's like the other side of this conversation. When those are combined, you get the same symmetric key as was obtained for encryption. And since it's the same symmetric key, and since it's symmetric, that key will do decryption. And so the other property this has is that every single file that's encrypted under Sodinokibi is encrypted with a different and unique symmetric key. So the file level of granularity would allow somebody to provide one decryption key for one file, and that decryption key could not be used anywhere else. It's completely unique to that one file.

So now the question is, since we needed the private side of the system key, how do we get that? Recall that the private half of the system key was encrypted separately using the public halves of both the operator's and the campaign's keys. This means that the private half of either the operator's or the campaign's key can be used to decrypt the system key. Since the public half of the operator key is embedded in the Sodinokibi code, which is individualized per affiliate and cannot be changed by the affiliate, this is what provides the REvil operators with a per-affiliate backdoor into Sodinokibi, enabling them to unilaterally decrypt anything that any of their software is ever used to encrypt. So this provides the REvil operators with protection against any rogue affiliate who might violate their rules in some fashion.

And both of these approaches would decrypt an entire campaign. That is, either the use of the operator's private key or the campaign private key can decrypt the entire campaign. Yeah, exactly. We saw with the Kaseya attack that the attacking affiliate was offering to decrypt individual systems for the bargain price of only $45,000 in bitcoin. The way this would be done would be done would be by distributing a decryptor, giving someone you had attacked a decryptor that only contains a specific system's previously decrypted private key. That allows them to combine that private key with all of that system's files per-file public key which is in the header of every encrypted file, use Diffie-Hellman to obtain the symmetric Salsa20 key, and you decrypt the file.

So without an explanation like what I've just provided, a flowchart of Sodinokibi's cryptographic operations is quite daunting. And in fact, even with - now it's on the screen. Get ready, folks. Even with that careful explanation, Sodinokibi's design can be a bit overwhelming, as we can see in the diagram below. I stared at it. I understand every arrow. There is not a single thing there that is not there for a reason and that does not provide some functionality. So I've often said on the podcast that with the cryptographic tools we now have at our disposal as a bit of a toolkit, you can pretty much do anything you want. And these guys are unfortunately a malicious example of creating a very sophisticated, very capable facility.

So the bottom line is that whoever developed this beast several years ago - it first appeared in 2019 - really thought the problem all the way through. Unlike many of the less professionally designed, half-baked ransomware systems that we've seen, like where you can easily get a decryptor after analyzing the software, it's like, oh, they used the key My Bananas Are Yellow. So that's not going to be hard to decrypt. Unlike that, these guys did not make any mistakes that would allow for short-circuiting the need to pay for decryption. And in the process they have created, unfortunately, a powerful and a very efficient file-encrypting system.

MIKAH: All right. I have a question, then.

Steve: Yeah.

MIKAH: So in a broad sense I'm understanding what you're saying. Obviously not the nitty-gritty details here. But one of the things that you mentioned is that each individual, you know, it goes as far as each individual file is encrypted. So then say REvil comes after me, and they get me, and I give them the bitcoin that they ask for. Are they going to pass me encryption keys or decryption keys for every single individual file? Are they going to pass me a program that decrypts my stuff? Are they going to decrypt it for me and let me re-download it? How does the actual process work to return my files to the way that they were before? If every single file is encrypted, do I have to have a key for every single one of those now?

Steve: Okay. So what happens is you would pay the ransom. They would know who you are because you would have been a campaign. You would be at the campaign level that you were working with them. Either the affiliate who was probably the person who attacked you, or REvil, if like the affiliate went belly-up or disappeared or something, either of those two people hold a private key that will decrypt the private key in your system. So, okay. So that's the first thing is your system has a key, but it doesn't do you any good because it's encrypted in a way you can't decrypt it.

MIKAH: Oh, okay.

Steve: So they will decrypt that. And then they provide you with a decryption tool. The key to your question is that in front of your files that are encrypted is a header. So Sodinokibi adds a little header, doesn't have to be big, it's probably 256 bytes or something. In that is the per-file data that doesn't let you decrypt it unless you have the decrypted system key.

MIKAH: Okay.

Steve: So the decrypted system key plus the per-file header provides the per-file decryption key to decrypt that file.

MIKAH: And so it's kind of both you get a key that you need to decrypt your system, but then they also give you a program that's going to actually do the process of decrypting all those files...

Steve: Yeah, actually they make it a little easier. They simply give you a program that will do the decryption for you.

MIKAH: Got it.

Steve: Because they've done the work of taking your encrypted system key, decrypting it with their private key, and they then build that into a little program and say, here you go. Just turn this loose, and it'll fix you. And presumably it recurses through your entire directory tree in the same way and takes the header off of the front of each file, performs the elliptic curve crypto to obtain the Salsa20 key, the same one that was originally used to encrypt it, reapplies it to decrypt it, and you get your file back.

MIKAH: All right. That makes sense.

Steve: And then the last little piece, since I thought our listeners would appreciate having this, Acronis reverse engineered the exempted folders, exempted files, and exempted file extensions which Sodinokibi will not touch. And it's at the bottom of the show notes. It's on the screen right now. Anybody who's interested can grab the show notes. The last page of the show notes has the list of folders that they don't go into, the list of files that they never encrypt, and the file extensions that they don't bother encrypting just because they're not, you know, it's like .bin, .hlp, .drv, .bat, .msc, .lnk, .cab, those things, you know, that are not valuable user content. There's things you could get, for example, from another system, so why bother taking their time, slowing down. And, by the way, there's a gazillion of those, right, .nls, .sys, .dll. Oh, my god.

So they want to skip decrypting stuff that they don't need to because they want to get the stuff they do need encrypted, encrypted as quickly as they can. So it's not like they're doing us a favor by leaving a bunch of stuff unencrypted. It's just there's no point because that stuff is all readily replaceable from another machine.

MIKAH: And part of the reason for that efficiency you were talking about earlier, working at the low level, is so that they can do it before they're detected? Is that the point of that?

Steve: Right.

MIKAH: So you wouldn't even notice your fans spinning up on your system as it's doing this encryption in the background?

Steve: Right. And in fact, what everyone is told, like when the first instant that ransomware is detected in a company, like an emergency broadcast is set, like unplug your computer now. We don't care what you're doing, pull the plug now. It's just like, to just stop it from spreading or going any further.

MIKAH: That makes sense. All right. Well, it was complicated, but we got there in the end.

Steve: Yeah. I think it made sense.

MIKAH: It does, it does. I'll have to read it about 15 more times to truly grasp it, but I'm glad you...

Steve: There'll be no test on this.

MIKAH: Oh, good. Okay, good. Whew. Thank goodness. Well, I think that brings us to the end of this episode of Security Now!; right?

Steve: Yup.

MIKAH: All right. Well, of course, Steve, thank you for all of your awesome work on this show. Folks can tune in live every Tuesday at 4:30 p.m. Eastern, 1:30 p.m. Pacific, 20:30 UTC to watch the show record live. Or you can go to TWiT.tv/sn where you can subscribe to the show on Apple Podcasts, Google Podcasts, Pocket Casts, Spotify, all the places. We try to be in those places, as well, so you can learn about the next PrintNightmare when it pops up. I don't know, does Leo usually - you've got anything you need to plug or want to talk about?

Steve: No, we're good.

MIKAH: All right. Well, then, that does it. And thanks for having me join you this week. Appreciate it.

Steve: I'm glad to have you fill in. Thanks so much, Mikah. Bye.

MIKAH: Bye-bye.