The Massive MOVEit Maelstrom

Description: This week, two big stories dominate our podcast. We start by taking a quick look back at last week's Microsoft Patch Tuesday. Then we examine the latest surprising research to emerge from the Ben-Gurion University of the Negev. What these guys have found this time is startling. Then, after sharing some feedback from our listeners and a long-awaited big SpinRite milestone announcement, we're going to spend the rest of our available time examining the story behind this month's massive cyber-extortion attack which is making all of the recent headlines and causing our listeners to tweet: "I'll bet I can guess what you're going to be talking about this week." Yes, indeed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-928.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-928-lq.mp3

SHOW TEASE: Coming up next on Security Now!, Leo's off to Disneyland. So I, Jason Howell, am filling in for Leo today, joining Steve Gibson who has all the words. Actually, a few very key security stories. One takes a look at what kind of security information can be gleaned from a tiny little LED power light. You might think not a lot. I'm here to tell you you're wrong. So that's coming up. Also this month's major cyber extortion attack, the MOVEit maelstrom. Steve goes into some intricate detail on that. Plus last week's Patch Tuesday. And Steve has a major milestone to announce with SpinRite. All that and more coming up next on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 928, recorded Tuesday, June 20th, 2023: The Massive MOVEit maelstrom.

It's time for Security Now!. I'm Jason Howell, filling in for Leo Laporte. I promise never to sing the theme song again. I'm sorry I did that. Right off the top I have regrets. Here joining of course the man that you checked out this podcast for, Steve Gibson, to talk all about security today. It's great to be here with you, Steve.

Steve Gibson: Great to have you. This was a surprise to me, but apparently not to anybody else, so...

JASON: You know, Leo just sometimes on a spur of the moment decides, I'm going to skip town and go to Disneyland. And that might sound like a joke, but that's exactly what's happening.

Steve: I don't think he's actually ever done that before. No, I think this is a first in 18-plus years. But, you know, what the heck?

JASON: Hey, when you own a company for 18 years, you might as well have the ability to take off.

Steve: That's a good point. I've done my own share of skipping out of my own company. So this week, it was funny because you got the show notes, and you thought, uh, did I get incomplete show notes? Is this, like, everything? It's like, well, yes. Two big stories dominate our podcast. We start by taking a quick look back at just - this is not one of the big stories - at last week's - fortunately it's not one of the big stories - last week's Microsoft Patch Tuesday. But then we're going to examine the latest surprising research to emerge from the Ben-Gurion University of the Negev. What these guys have found this time is startling. And I'll give a little more back story when we get to that. But then, after sharing some feedback from our listeners and a long-awaited big SpinRite milestone announcement, we're going to spend the rest of our available time examining the story behind this month's massive cyber extortion attack, or I'm not sure if you'd call it an attack or attacks, plural, because it's so big that it's been making all of the recent headlines, and it's been causing our listeners to tweet to me things like "I'll bet I can guess what you're going to be talking about this week."

JASON: You probably get that all the time.

Steve: Yes. It's often, pretty much, pretty clear.

JASON: Yeah.

Steve: So anyway, I titled Episode 928 for this June 20th, titled it "The Massive MOVEit Maelstrom" because this is really bad. And so we'll spend some time looking at what happened, why it happened, how it happened, and what does it mean about the future.

JASON: All right. Well, we've got a lot to look forward to. I'm super curious about that. The two big stories...

Steve: Oh, and of course we have a Picture of the Week.

JASON: Which if I sounded distracted is because I was looking at it. I was trying not to laugh in advance because it's pretty awesome.

Steve: Isn't it wonderful? Oh.

JASON: It's pretty awesome. And we can finally talk about the Picture of the Week, which had me kind of stop in my tracks a few minutes ago, trying to figure this out. But tell the audience about it first.

Steve: Okay. So there's a little bit of back story since you're not always the host of the podcast.

JASON: Sure.

Steve: There was an earlier famous bicycle lockup Picture of the Week which really made you wonder because they had locked their bicycle with a big, heavy security band to a post that had nothing on the top of it. I mean, it was just a post.

JASON: I love that.

Steve: So if anyone had wanted the bicycle, they could have just lifted it off of the post, you know, I mean, lifted the lock off. Anyway, so this one is great, and I'm sure one of our listeners did send it to me knowing that we have an affection for wacky pictures. I gave this the caption, "If it's not tied down..." because, if we believe this, I mean, this could obviously be set up, but still it's great. Somebody really, really, really wanted to lock up their bicycle.

So they had this seriously strong metal steel gate that has a bar running along the bottom, and one running along the top. And so they used those U-shaped strong bicycle locks through both the front tire and the back tire, one each, looped around the steel rail that runs along the bottom. And just, I mean, you would imagine that would be enough. But they figured, oh, no, I want to also lock something else. And since there's a bar running across the top, they thought, well, I'll run another security cable underneath the seat through the metal mounting of the seat to further lock down the top of the bike.

What we see in this picture is exactly those three things, and only those three things. Someone stole the bike out from, like the chassis, out from the three things that were locked down.

JASON: The frame's gone. No more frame.

Steve: The frame is completely, that's true, frame is gone. There's just a front tire and a back tire, right where they always were, locked rigidly to the lower rail, and a seat hanging from the upper security cable on the top rail, and nothing in between. So those three pieces are disconnected from each other because, yes, the frame of the bicycle, apparently someone thought, okay, I'll show you, and it's gone.

JASON: I guess things could have been worse. You could have lost the whole bike. Instead you just lost the frame. I'm looking at it. And so what confused me earlier was when I was looking at the seat, it kind of looked to me like the seat was just hanging on the loop. But what I realized as I looked further, there is kind of like a metal thing underneath the seat that it's going through. So that seat isn't going anywhere.

Steve: Correct.

JASON: I'm curious to know, like, what's the most expensive - maybe I'm thinking too deeply into this. What's the most expensive part of the bike? Is it the frame? Is it the tires and the seat? Because if it's the frame, I mean, you know, whoever took that bike obviously knew what they were doing. They're like, well, still, I mean, I realize they don't me to take this, but I can. And that's still worth something.

Steve: Now, I can offer some guidance, Jason, about our Pictures of the Week.

JASON: Yeah, don't think too deeply?

Steve: It is not possible to think too deeply about them. Leo and I have sometimes spent a good half hour on why is there a gate in the middle of this field?

JASON: Yeah, that's good. So you don't have to worry about overthinking this or thinking, you know, like going in too deep. No. I don't think it's possible to go too deep into any of this podcast's Pictures of the Week. Fortunately, it's not possible to go too deep into last week's Microsoft Patch Tuesday. This being the third Tuesday of the month, we're able to look back on last week's Patch Tuesday.

It's still, to me, somewhat astonishing that just last week Microsoft patched 26 remote code execution vulnerabilities. You know, when Leo proposed this podcast to me, I didn't have any idea that we'd still be going strong after 18-plus years. There's no sign, similarly, that Microsoft is ever going to run out of serious vulnerabilities. The only thing, the only way to explain this is that they're, you know, everything they do is creating new ones, and so they're fixing some of the old ones, but at the same time creating new ones to be found in the future. So similarly, I don't think we're in any danger of Microsoft running out of serious vulnerabilities.

Four of these 26 RCEs, remote code execution problems, were critical, and three of those four were spotted and fixed in a single Windows component known as PGM. That's the Pragmatic General Multicast queue. Because, you know, the thing you really want from your general multicast queues is pragmatism. What you don't want from them is remote code execution flaws, especially when they're bearing a CVSS score of 9.8. So after applying last Tuesday's patches, for the time being there will be three fewer RCEs, which probably makes the decision to use the queuing API in the first place somewhat more pragmatic. So that's good.

There were also 17 elevation-of-privilege problems fixed, only one of which made it to the critical list. It had a CVSS score, however, of a whopping 9.8, which, you know, I think maybe we've seen a 10 once or something. It's got to be, you know, the really bad ones are 9.8, and they save the 9.9s and the 10.0s for, like, the end of the world. So we've only had the end of the world happen I think once before. Anyway, if exploited, an attacker could gain administrative privileges as a consequence of one of these 17 elevation-of-privilege problems.

Microsoft wrote: "An attacker who has gained access to spoofed JSON" - that's not you, Jason, that's JSON - "Web Token authentication tokens could use them to execute a network attack which bypasses authentication and allows them to gain access to the privileges of an authenticated user. The attacker needs no privileges, nor does the user need to perform any action." Now, since we're talking about, and I didn't mention this yet, the SharePoint server, which is the same sort of database sharing that has landed the MOVEit Transfer software which we'll be talking about in the second half of this podcast, into such hot water earlier this month, getting that one patched is something that should probably not be put off, especially given that it's a 9.8. Microsoft is saying, yes, we're glad we fixed this, and we're going to try not to create any more.

Overall, of the 73 flaws fixed, six are rated critical, 63 are important, two are moderate severity, and one is low. So included within that group are three issues which were fixed in the Edge browser. It's worth noting that during the intervening month between last month's patches and this month's, Microsoft did also eliminate 26 other flaws in Edge, included among those a zero-day which Google last week disclosed which was being actively exploited in the wild. So that's fixed. But this month is the first time in several months when none of the known and patched problems were either publicly known or under attack at the time of their fixes. So that's good.

And we mentioned both publicly known or under attack because Microsoft has their own definition of zero-day. Everybody else considers a zero-day to be one where it's first learned about because it's spotted being used in an attack. Microsoft broadens that to say, if they're informed of something they didn't already know about, then it's a zero-day, which, I don't know, seems overly pessimistic to me. But anyway, we didn't have any.

The final note is that two of the remaining remote code execution vulnerabilities were found and fixed in Exchange Server. Being that Exchange Server is all about exchanging content with the outside world, it's generally a significant portion of an enterprise's attack surface. So absolutely want to get that fixed.

Okay. So I got that little bit of housekeeping out of the way. Next is one of two big pieces of amazing stuff. I titled this "Does EVERYTHING [in all caps] leak?" This next bit of jarring news leads me to pose that question; you know? Does everything leak, as in leaking information?

A couple of days ago, I received a Twitter DM from Ben Nassi. Ben tweeted: "Hi, Steve. My name is Ben Nassi, a postdoctoral researcher at Cornell Tech and a long-time listener of Security Now!. I just published a new research that I think you should see. We

recovered cryptographic keys from devices by obtaining video footage of their power LED. The devices were not compromised," meaning ahead of time. He says: "The research will be presented at Black Hat and Def Con this year." And then he sent me a link to their research PDF.

Okay. So the summer is approaching, with Black Hat and Def Con being held back-to-back every summer, as we know, in Las Vegas, Nevada. Ben also sent a link, as I said, to the research paper which is titled "Video-Based Cryptanalysis: Extracting Cryptographic Keys from Video Footage of a Device's Power LED."

JASON: That is crazy.

Steve: And I can still hardly believe this even as I'm reading it.

JASON: I can't believe that. That's pretty insane. I'm super curious.

Steve: You're going to love the details. So first of all, again, because you're not always the host of the podcast, my mentioning the Ben-Gurion University of the Negev may not, like, ring any bells for you. It does for our listeners. Ben is one of the prolific researchers there who have for years brought us an unending stream of often entertaining, but also often sobering, extremely clever examples of data exfiltration. Our longtime listeners will recall, for example, their work with extracting audible conversations from a room at a distance by visually detecting the sympathetic vibrations induced in a birthday party balloon, the leaf of a plant, or a light bulb, all of which allowed them to recover the audio filling the rooms occupied by those objects. I mean, and they've done all kinds of things.

JASON: Wow.

Steve: I think, yes, it's just incredible. We could safely label these guys the masters of the detection and recovery of side-channel information leakage. And so now we have another. And whereas some of their schemes have required malware to first be installed in the victim device, which is why he was careful in his tweet to me to point out not this time, you know, like for example they've deliberately changed the sound being emitted by a machine's power supply and used that for remote signaling. Or changing the system's fan speed in order to signal an air-gapped microphone.

You know, and for example, when they do that, it's easy to say, oh, change the system's fan speed in order to signal an air-gapped microphone. When they do this, though, they look at how long it takes for the speed to change when they change it, and how many discrete levels of speed they can make distinguishable by a microphone at what distance, because the number of levels and the speed it takes to get to among each level determines the bit rate at which they're able to send, I mean, these guys wrestle this stuff to the ground all the way, every time.

So, okay. In order to put myself in the same place as our listeners as I share this, I have not yet even glanced at Ben's paper. I have now because I wrote the show notes. But as I was writing the show notes, as I was first putting this down, I had not opened the paper. So we're going to do that together in a moment when I share the paper's abstract, which as I said, I have not yet seen as I'm writing this. But what we immediately and intriguingly ascertain from Ben's tweet is that, astonishingly, variations - and again, we're deriving this from everything we've learned over 18 years of the podcast; right? Variations in the work being done inside devices, where that work is dependent upon cryptographic secrets, must be sufficient to produce tiny variations in the power being supplied to such a device's LED power indicator.

Now, that's surprising enough, that any such fluctuations would be so tiny as to be detectable. And that they are theoretically detectable at a distance by a standard video

system or, you know, despite the depth of digitizing resolution, the effects of video compression, or the frame rate of the video, right, which is going to be limited. But to do what they've done, Ben's group apparently overcame all of these practical barriers, and they've done this before. And they're presenting this at Black Hat and Def Con. So, yeah.

And as I've observed in the past with their work, what really distinguishes their accomplishments is that they solve the whole problem. They just, you know, the headline on the paper tells you where they're headed, but where they go is typically amazing. Okay. So now let's discover together what Ben's group explains at the start of their paper.

They write: "In this paper, we present video-based cryptanalysis, a new method used to recover secret keys from a device by analyzing video footage of a device's power LED. We show that cryptographic computations performed by the CPU change the power consumption of the device, which affects the brightness of the device's power LED. Based on this observation, we show how attackers can exploit commercial video cameras." And they said: "For example, an iPhone 13's camera or Internet-connected security camera, to recover secret keys from devices. This is done by obtaining video footage of a device's power LED, in which the frame is filled with the power LED, and exploiting the video camera's rolling shutter to increase the sampling rate by three orders of magnitude from the frames-per-second rate of 60 measurements per second, to the rolling shutter speed of 60K measurements per second, in the iPhone 13 Pro Max."

Okay. So I'm going to pause here for a minute and say that's brilliant. With an LED, even if the illumination across the surface of the LED is not perfectly uniform, as in fact they aren't, being a solid-state illuminator, any change in illumination will be uniform and effectively instantaneous. So this is the key that allows them to obtain a sufficiently high effective sampling rate from an otherwise grossly insufficient 60 frames per second video recording.

So continuing, they said: "The frames of the video footage of the device's power LED are analyzed in RGB space, and the associated RGB values are used to recover the secret key by inducing" - and I'm sure that's a typo, they meant deducing - "the power consumption of the device from the RGB values."

Okay, now, pausing again for a moment, we long ago talked about variations in power consumption during cryptographic operations being a well-understood side-channel that could theoretically be used to reverse engineer the work being done by a device when that work is a function of secret data. But the presumption has been that that theory runs smack up against reality when there's no practical way to obtain instantaneous power consumption measurements without hooking deeply into a target device's electronics. These guys have quite cleverly solved the problem of doing that. They realized that minute variations in the device's power draw would induce tiny changes in the system's supply voltage.

The instantaneous brightness of an LED is determined by the instantaneous current flowing through it. LEDs will invariably have a resistor in series with them to set their operating current at the given supply voltage. But that current is not otherwise regulated. And that means that any variation in the system's total supply voltage will create a variation in the LED's current and therefore in its illumination. I'm still surprised this works, but apparently it does. And you have to imagine that the world's intelligence services have just perked up in response to this news.

So they finish their abstract, saying: "We demonstrate the application of video-based cryptanalysis by performing two side-channel crypt-analytic timing attacks and recover, first, a 256-bit elliptic curve DSA key from a smart card by analyzing video footage of the power LED of a smart card reader via a hijacked Internet-connected security camera

located 16 meters away from the smart card reader; and, two, a 378-bit SIKE key from a Samsung Galaxy S8 by analyzing" - get this - "video footage of the power LED of Logitech Z120 USB speakers that were connected to the same USB hub that was used to charge the Galaxy S8 via an iPhone 13 Pro Max. Finally, we discuss countermeasures, limitations, and the future of video-based cryptanalysis in light of the expected improvements in video camera specifications." Oh, my lord.

Okay. So before I go any further, I'll just mention, like in terms of extending this, now that we know that lights are flickering or just barely changing their illumination, the Light Pen which I developed back in 1983 for the Apple II had a response time of 140 nanoseconds because that was the pixel clock rate of the Apple II's video. 140 nanoseconds is a rate of 7.14 MHz. Now, that happens to be twice the NTSC color burst frequency of good old NTSC video back then, and that was part of Woz's design brilliance for the Apple II.

Okay. But my point is, since monitoring the power LED of devices that are performing secret computations has now been proven by these guys to work, it would be trivial to take the technology of a high performance light pen, place its photodiode at the user end of a telescope, and aim that scope at any power LED of any device containing secrets to begin collecting data. The data gathering and secret gathering power of such a system with 7 MHz bandwidth is somewhat terrifying.

Okay. So while it would be possible to create a highly sophisticated spying scope, the brilliance of what these guys have discovered and invented is the ability to use existing camera technologies, off-the-shelf tech, thanks to their observation that the cameras in our devices do not actually snap an entire scene at once. Instead, they actually scan the image from top to bottom or left to right in much the way that the images of our original cathode ray tubes did. This brilliantly allows them to sample the illumination of a device's LED with far greater temporal resolution than the camera's overall frame rate. Here's how Ben's paper describes it under the title "Increasing a Video Camera's Sampling Rate Using a Rolling Shutter."

They wrote: "We note that the FPS [frames per second] rate supported by the vast majority of commercial smartphones and security/IP video cameras is limited to 60-120 frames per second, which is sufficient for performing cryptanalysis. In order to increase the number of measurements per second sampling rate to a level sufficient" - oh, I'm sorry. I hope I said 60-120 frames per second which is insufficient for performing cryptanalysis. Of course. That's just not enough, I mean, the computations happening inside are way faster than that. So that's not going to cut it.

So they said: "In order to increase the number of measurements per second, the sampling rate, to a level sufficient for cryptanalysis, the attacker can exploit the video camera's rolling shutter. The rolling shutter is an image-capturing method in which a frame of a video in video footage is captured by scanning the scene vertically or horizontally. When this method is used, a frame per picture is not actually composed of a single snapshot of a scene taken at a specific point in time, but rather is composed of multiple snapshots taken of vertical or horizontal slices of the scene at different times.

"With a vertical rolling shutter, a sensor's pixels are exposed and read out row-by-row sequentially at different times from top to bottom, or left to right, according to a configurable shutter speed which determines the amount of time that the sensor is exposed to light. Because each row, or a group of adjacent rows, in a sensor with a rolling shutter is captured at a different time, attackers can increase the effective sampling rate from the camera's frame-per-second rate of 60 or 120 frames per second to the rate at which rows are recorded, a rate which is based on the shutter speed."

Okay. So, wow. The biggest limitation is that for this clever rolling shutter rate up-sampling to work, the LED's image, as I briefly touched on at the start, the LED's image must fill the entire camera frame, right, because you want all of the individual row or line samples to be of the LED. So it can't be just a tiny spot in the middle of the camera. You've got to zoom the camera in so that the LED's light fills its camera sensor. But there are tiny external lenses that can be added to a smartphone to make that easier, and there are doubtless many applications where the installation of a device, such as a smart card reader, is made with the inherent assumption that its internal secret cryptographic computations are not being broadcast outside of the device. Ben's latest work has, amazingly enough, shown the world that this longstanding assumption has always been wrong.

You know, we've often commented about the fact that the Internet "activity LEDs" which are ubiquitous on all networking equipment, routers and switches and everything, do not reveal anything about the data that's passing through their interfaces. Well, it turns out we've been looking at the wrong LED.

JASON: I'm thinking of the phones that I've had, and of which I've had a lot over the years, and how many of them have some sort of an LED light. Like that used to be a really common thing. It used to be used for like notifications and whatever. And if you plugged it in to charge, that little LED would light up. But I don't feel like I see that as much anymore. So at least there's that obscurity element on phones, let alone other devices.

Steve: Correct. Correct. I mean, yeah, on phones. And in fact, as I mentioned, in one of their documented attacks, they put a phone virtually on the LED of a smart card reader while it is scanning the smart card. And they don't have to process it at the time; right? So all they have to do is they just turn the recorder on, and they're just recording the video while the smart card gets scanned, and they're able to get the private key of that smart card.

JASON: Yeah. Wow.

Steve: Oh, goodness.

JASON: How smart are people that they can be like, well, I wonder, you know, I have a hunch, and they look into it, and they're able to figure this out. I'm in awe of the ability to do that.

Steve: Yeah. This was a beautiful piece of work. So Ben, I know you're listening. Congratulations.

JASON: Wow.

Steve: Wow. Just a fabulous piece of work.

JASON: Back to you, Steve, with Closing the Loop. Lot of feedback today.

Steve: Yeah. So Kevin, tweeting from @sharpestmarble, he said: "Listening to Security Now! 927, and you're praising Apple for Live Voicemail, which they're going to be developing. But this is something Google phones have had for over a year now." And Jason, as I said before we began, I was glad to have you here since you know Android, and I don't. So is it all Android phones, or Google Pixel devices?

JASON: Well, I mean...

Steve: Like Samsung?

JASON: Yeah, yeah, definitely Samsung has the ability to do it, too. I'm not sure if they're tapping into the same thing that Google can do. But the visual voicemail, transcribing voicemail that comes in, definitely Google, I mean, Google's had Google Voice technology for more than a decade. I don't even know when Google Voice was bought by Google. It was Grand Central prior to that. And somewhere along the line they started to automatically transcribe the voicemails. I think that was kind of the beginning, if I'm not mistaken, the beginning of Google's kind of experience in doing this live transcription of these voicemails, and they definitely got better with it over time.

And then that feature has expanded outside of just voicemails into, you know, there's a feature that I was telling you about prior to the show that's embedded into the volume rocker on Pixel devices and some other Android phones. And it's basically a live transcribe of any audio that's passing through the phone. So that could be a video you're watching. That could be the music, you know, if there's any speaking in music, it would transcribe that. The phone call, all of this stuff is transcribed in a little closed caption box that you can drag anywhere on the screen. Google's gotten really, really good at it, and it's one of my favorite kind of innovative features that Google's integrated in recent years. But, yeah, Google's solid on this stuff.

Steve: Thank you, Kevin, for letting me know so we could also give credit where it's due. Not a great innovation, Apple, but you're catching up.

JASON: They're all borrowing from each other. You know, one side will get a great idea, and they'll implement it, and then suddenly that becomes table stakes, you know, when they all have...

Steve: That's right, wasn't Apple the first fingerprint reader? I think Apple had the first fingerprint reader.

JASON: Could be. I'll say off the top of my head I can't remember. But, you know, they all borrow from each other.

Steve: Yeah. So Vincent Stacey, he said: "Think Apple could use the same on iPhone technology protecting children from unwanted images to catch and analyze the iMessage vulnerability?"

So a couple weeks ago we talked about an iMessage vulnerability which was probably impossible to catch because the end-to-end encryption protected the contents of the message, which, once it launched malware, deleted the attachment which iMessage brought along with it, thus removing all of its fingerprints of the attack. And so I made the point back then, the observation that the same really strong encryption which is protecting the content of Apple's iOS and iMessage user messaging is also protecting the bad guys who have an exploit of something in the system. If it deletes itself, how is anyone ever going to be able to analyze it cryptographically or forensically?

Anyway, in answer to Vincent's question, I suspect that the same technology won't be useful because one aspect, which is scanning for unwanted material, is within the bounds of normal operation; while the operation of this malware is explicitly out of bounds. The trouble with catching it is that whatever it is that the iMessage vulnerability is doing, it's breaking what are supposedly unbreakable rules. It is somehow escaping from within the rigid controls and sandboxing that Apple has explicitly and deliberately erected around it to contain and neuter exactly such exploits. We know only that the malicious iMessage is bringing along an attachment.

The fact that this is a zero-click exploit means that for the benefit of its user, iOS must be automatically attempting to render whatever it is that iMessage has brought along with it, and that it is during this auto rendering that iOS loses control. So it's breaking the rules that iOS is using, so I can't see how iOS attempting to analyze it within the bounds of the rules would succeed.

Okay, but having said that, if iOS were somehow able to record all iMessage attachments before performing any processing of the attachment, and that was in an immutable audit log on the target device which could not be, being immutable, that log could not have anything deleted. Even if the attachment were deleted, the record of it would be preserved. Then, if a target device was to be compromised, and if that compromise was recognized, then it would seem possible for a forensic analysis after the fact to be made.

You know, this theoretical immutable log would need to be immutable even to Apple's own software; right? Because, if not, then malware running as Apple software could delete it. But if there were something which, once written, no software running in an iOS device could erase it, then it's certainly feasible, given all the other preconditions. One of the problems is that the target needs to know that they've been a victim. Probably often they don't. And if they knew it, then they would have to know that they could take their phone to someone and say, hey, I've been victimized. I need the immutable log, the immutable forensics log, which we would then all know that iOS devices had, to be analyzed. And that's a problem, too; right? Because that would say that every attachment was being recorded by the iPhone. And a lot of people don't want that. So probably no good way to solve this problem, unfortunately.

Jason Egan said: "Hey, Steve. After hearing about the grc.sc situation, and please forgive me if I'm late to giving this option," he said, "I wondered if you couldn't just solve the problem with a RewriteRule on your host? I've had to do this in the past with some of my projects."

Okay, now, so just to remind everyone what Jason's talking about, on Episode 926, two weeks ago, I ended the podcast with a shortcut of the week, grc.sc/926, you know, .sc as in shortcut; right? And then the next week I commented, Mrlinux11 wrote saying: "Hey, I tried to use that, but I'm getting a page not found." Well, it turned out that he was using www.grc.sc. And so something had added "www." in front of the unencumbered, simple "grc.sc" and broken it.

So, now, Jason is saying you could do a RewriteRule. Okay. As I've mentioned in the past - well, okay. So first, so he's saying that I could solve the problem with a RewriteRule if my DNS included *.grc.sc in addition to grc.sc. And that's true. It never occurred to me that anyone would stick an arbitrary www. in front of grc.sc. As I've mentioned before, I do have a *.grc.com in my DNS since that's able to handle all of the many prefixes that I use, like "www.", "forums.", "dev.", "sqrl." and whatever else .grc.com.

And I do redirect any other wayward queries over to www.grc.com. Many years ago, GRC's web servers accepted connections on either grc.com or www.grc.com. I figured, why not? But then we noticed that Google's search results were coming up with a mixture of either grc.com and then whatever URL, or www.grc.com and whatever URL. Google was seeing these as separate distinct websites and indexing them as separate websites. And people were linking to GRC somewhat arbitrarily as either grc.com or www.grc.com. And that was having the unintended side effect of reducing our Google page ranking by diluting the number of incoming links among both sites, what Google thought was two different sites.

So I changed GRC's web server to redirect by returning an HTTP/301 redirect from any plain grc.com over to www.grc.com. And Google's spiders quickly learned that this was a single site which could, yes, be accessed by two different domains, but they all brought

you to www.grc.com. And then that consolidated all the links around a single domain, and that worked. At the time I had to choose one or the other, and I'm really not sure I did the right thing by choosing the longer "www.grc.com" as the enforced default. Yes, it's technically more accurate of the two. But whenever I or anyone talks about GRC.com we say "grc.com," not "www.grc.com." And I'm sure everyone just enters "grc.com" into their URL bar whenever they want to visit.

So anyway, in the next tweet, the answer to how this www.grc.sc came about was provided. A guy tweeting as CPUGuru, he said: "I found the source of the errant 'www' that you discussed in the podcast. The hyperlink in the 926 PDF actually includes it." So first of all, thank you, CPUGuru. Good observation. And it was Google Docs that did this. I typed in grc.sc and /926 and hit ENTER. All I saw was grc.sc/926. But sure enough, if you hover your mouse over that link, there's a www. in front of it, which Google Docs silently added, which I didn't ask for. So anybody who clicked the link in the Docs as I invited them to, I mean, I made it bold and increased the font size, saying, "Here, click me." And that didn't work, thanks to Google Docs.

Jos Javier Vegas tweeted: "Hi, Steve. Regarding SN-927" - so that was last week. He said: "There is no official Let's Encrypt client today. They transferred their implementation to the EFF, and it's now called 'Certbot,' which is the only one they recommended."

So thank you, Jose. That's good to know. I am still issuing certs for my own servers the old-fashioned way, through my favorite Certificate Authority, DigiCert. But assuming that the world is going to switch, as has been proposed, to 90-day maximum life certificates, I'm glad that DigiCert also supports the ACME protocol. And I will certainly then need to find automation for this, as will pretty much everyone because nobody wants to be generating new certs every three months. That's way burdensome.

And finally, Michael Horowitz, tweeting from his Defensive Computing Twitter account. He said: "Steve, an FYI about HP+ printers." He said: "They must be online all the time, even if connected to a PC via USB. And you must have an HP account." He says: "Perfect for spying. Details on this page." And he sent a link. Michael has a bunch of really great pages. This one is at the domain defensivecomputingchecklist.com. And so it's defensivecomputingchecklist.com/printers.php. And it's interesting that it's .php because we'll be talking about URLs ending in scriptable languages not long from now.

And so there was a perfect example of printers.php, where that's actually running a PHP script which is then pulling the contents of the page from some sort of CMS, you know, Content Management System. Anyway, the first line of Michael's printers page reads: "I hate printers. So, too, does Leo Laporte, who is known as the Tech Guy on the radio. He will not take phone calls about printers." So that's Michael's lead-in to his page about printers.

I read through some of what Michael wrote on that page, and I recommend it to any of our listeners who may be curious to know more, and who might be in the market for a printer. Michael has collected many reviews and anecdotes, and he tells a horror story of spending half a day trying to install an HP printer/scanner for a friend. Mind you, Michael knows his way around PCs as well as any of us, yet he repeatedly hit wall after wall just trying to, like, play by the rules and install HP software. At one point he mentioned that he downloaded the first package, which was 310MB, which, again, .3GB for printer software?

Anyway, his experience further supported my earlier statements when we were talking about this debacle with the HP printers, that I have long found HP software to be unconscionably atrocious. It is just the worst. And short version, apparently Brother

makes the printer that most people recommend and have no problems with. I think it's Brother.

JASON: Yeah, it is Brother. It's funny that you even mention this because I've been going through, it's not an HP printer, but it's an Epson printer at home that we've had, this Epson Inkjet forever. And this is the third time that I've had to take it apart and try and repair it and get it back to working and everything. And it didn't go back together the way I expected it to. I was like, you know what, I'm done. Like I actually just talked to Leo when he came in today. I was like, what - send me the make and model. He's like, "It's Brother. It's a LaserJet printer." He's like, "Get it. You won't need another printer."

Steve: That's exactly the same conclusion that Michael came to.

JASON: Yeah. So I'm getting that. I don't know which one it is. I have to look up the model number and get it from him. But I'm not going to look back. I'm just going to do that.

Steve: Good.

JASON: Yeah, and all printer software sucks. It's just so bad. It's, ugh.

Steve: What happened?

JASON: I don't know.

Steve: It's such a simple thing to do.

JASON: Totally.

Steve: You would think. But...

JASON: It's just always horrible.

Steve: Wow.

JASON: That's been my experience, anyway.

Steve: Okay. So finally, it is with no small amount of pride, a feeling of accomplishment, and some pent up relief that I can finally assert that, as far as I know, the work on the business end of SpinRite 6.1 - which is its DOS executable side - is finished. There are presently no remaining known bugs, great or small. Last Sunday afternoon, so two days ago, I announced the availability of the 29th alpha release to GRC's 696, but I noted we just got one more, so now we're at 697, registered SpinRite 6 owners who have been testing 6.1 since we began the alphas in November of last year. And I suggested for the first time to the group that the code we all now have would likely and certainly could, with probably very few remaining changes, be moved into beta status to soon become SpinRite 6.1's shipping code.

6.1 contains a great many new features, one of them being an integrated FAQ which explains the choice of SpinRite's five redefined operating levels, its many command-line options, and a bunch of other useful tidbits. Sort of, you know, power user tips and things that we've discovered and things to try and so forth. So while the dust settles on this latest alpha, I'm currently writing that FAQ. And that'll give some time for any testers who may have become bored with the seemingly endless interim development releases to give this proposed final code one last check.

Once we have the finished DOS code, that code, along with the very first work I did on the project, which was to create GRC's InitDisk USB drive prep utility, all that will be integrated into an updated SpinRite Windows app, and then we'll have 6.1. Any new purchasers of SpinRite will automatically then begin receiving 6.1. But since I want to let all of this new code breathe a bit - like any nice red wine that you open, give it a little time to breathe, it'll mellow - I plan to hold off on announcing it to all of SpinRite's past purchasers until it's had a bit more time among a wider audience.

So as soon as the official upgrade path is established, I'll be inviting all of this podcast's listeners, many of whom have previously purchased SpinRite, to update to v6.1. Then, once it appears that it's going to be clear sailing, I'll begin the process of informing everyone who owns 6.0 that they get a free upgrade. Even though it's been 19 years, it's free for them. So no new action needs to be taken by anyone who hasn't already jumped onto testing the pre-release code. I just wanted to note that the project had achieved a significant milestone along the way.

JASON: Big-time. Congrats.

Steve: Thanks to everybody for their support along the way, all this time. And thank you, Jason. Yes.

JASON: Yeah, good work.

Steve: It's like, ahhh.

JASON: Yeah, I bet. Sigh of relief. You only get a week, though, to rest on your laurels. And then you've got to start working on the next version. I'm sorry, Steve.

Steve: That's it. That actually is the plan. I'll be starting on SpinRite 7 immediately.

JASON: I was going to say, just give yourself a week, I mean, you deserve to take some time, maybe go to Disneyland is an idea.

Steve: So inevitably some of the testers have been asking for this or that new feature. I mean, like extensions beyond what it currently is. And I've said, look, I said, understand this. My goal has been to update 6 to 6.1, to resolve the problems with speed, the problems with drive size. And inevitably I ended up doing way more than that. But I said, "Understand this. My goal is to obsolete 6.1 as quickly as I can with 7.0 because there were some things we could not do. We could not do UEFI. DOS will not run on UEFI. Period. So in order to move to UEFI, we have to leave DOS."

Well, that's a big change. So that's not like an incremental - no one would call that an upgrade. And also I wasn't able to do native support for USB. So USB still runs through the BIOS, as all USB always has for all of SpinRite. But I want to get to 7.0 so I can resolve those things also. But it was just necessary to say, no, we're stopping at 6.1 and after that point my goal is to obsolete it as quickly as possible. And once everyone has 6.1, that'll be what they want, too. So it's a win-win.

JASON: Absolutely.

Steve: And Jason, let's do our last sponsor insert, and then we're going to talk about the Massive MOVEit Maelstrom.

JASON: And now it's time to move it. It's time to get into the Massive MOVEit - is it Maelstrom? I always see that word, and I'm like, is it Maelstrom? Maelstrom?

Steve: Maelstrom.

JASON: Maelstrom. I think it's Maelstrom.

Steve: Think so.

JASON: Maelstrom, okay.

Steve: So our main topic today arrives about three weeks after the first signs of this significant problem arose. I've been aware of and watching what's been happening. But it wasn't until this past week that the scope and scale of the problem became fully apparent. So this week we need to do a bit of catching up with what's been going on, and then we'll look at where we are today.

The trouble surrounds a globally popular file transfer facility named MOVEit. MOVEit has both a local version called MOVEit Transfer, and also a cloud-based solution, MOVEit Cloud, Basically it's a file-sharing and management solution from its parent company, Progress, www.progress.com/moveit, M-O-V-E-I-T. And the way they've spelled it is MOVE is all caps, and "it" is small. So MOVEit describes itself as "Managed File Transfer Software."

They said: "Secure File Transfer and Automation Software for the Enterprise. Guarantee the reliability of core business processes and transfer sensitive data between partners, customers, and systems the secure and compliant way with MOVEit. Secure, Auditable, Automated, and Compliant File Transfer - On-Premise and in the Cloud. MOVEit provides secure collaboration and automated file transfers of sensitive data and advanced workflow automation capabilities without the need for scripting. Encryption and activity tracking enable compliance with regulations such as PCI, HIPAA, and GDPR."

Okay. We've just read that it's compliant with PCI, HIPAA, and GDPR. Unfortunately, another abbreviation its web frontend is fully compliant with is SQL, and not in the way they intended. That's right. The industry has been hit with another very powerful and significant SQL Injection attack. I was tempted to title today's podcast "Little Bobby Drop Tables." And our longtime listeners will understand the history of that name, or cartoon, or joke.

The last time we were on this subject, I railed against the fundamentally broken design of the SQL command model, which exposes a fully capable command language to a web server that typically only needed to issue queries against the data. Yet the unrestrained nature of this powerful command-line interface meant that the web server could do anything it wished. And if someone could arrange to get the web server to pipe their own user-supplied text through to the backend SQL server, such a remote user could do pretty much anything they wished.

My denigration of SQL generated some pushback from some of our listeners who quite correctly noted that there were several other much safer and much more proper ways to do this with modern SQL servers. And those listeners were of course 100% correct. But I wasn't saying that safer and more proper ways had not since been developed to do this, but that the original unsafe ways also continued to be present in the interest of endless backward compatibility and not breaking legacy systems. Consequently, nothing prevents the original horrible and fundamentally insecure approach from continuing to be used.

And so here we are today in 2023, we have the latest example of this bad architecture striking once again. Even if someone was being as responsible as they could be with this hot potato, it would be the programmer's responsibility to try to think of all possible ways bad guys might attempt to sneak commands under the cover of data by encoding them strangely, using an unusual language locale, or who knows what?

As is always the case with security, the battle is asymmetric. The programmer must block every possible avenue of conquest, whereas the bad guys only need to find one way in. To Progress's credit - the publisher of MOVEit - from the start they have not tried to hide any of this in any way. Right there on their main product promo page they write, they have a big yellow, bright yellow warning, says: "PRODUCT ADVISORY: MOVEit Transfer and MOVEit Cloud Vulnerability. Click for mitigation measures and patch information."

Okay. So here's what we knew three weeks ago as May was coming to an end. As we already know, MOVEit's solution includes a web-based frontend to manage the sharing, uploading, and downloading of files. This makes sense in an era of JavaScript which is able to accept drag-and-drop uploads and manage local downloads. When we mix in web authentication, which is, as we know, an entire discipline of its own, it's quite possible to create a fully functional web-based file management and distribution system. And when the alternative was email, you know, using email for that, there's no competition. MOVEit also supports FTP, but I assume that's for legacy purposes since FTP support has, as we know, finally been deprecated and removed from web browsers, and it's unclear what advantage FTP has any longer over any modern web browser solution. Once upon a time, sure. Today, eh, not so much.

As for this vulnerability, the bad news is that it was a true zero-day. Progress learned of it only after and because the bad guys were already exploiting it. And boy, were they. Being responsible, Progress quickly patched all of their supported MOVEit versions and their cloud-based, you know, the MOVEit Transfer local, self-hosted versions, as well as their cloud-based service.

There are essentially four things that can be done with this vulnerability: the deletion of existing data, the exfiltration of existing data, the modification of existing data, and the implantation of new files and malware. As it turned out, two of those four have been seen happening.

Mandiant, which is now owned by Google Cloud, has been tracking the MOVEit breach activity under their uncategorized moniker UNC4857 and posted that the opportunistic attacks have singled out a wide range of industries based in Canada, India, the U.S., Italy, Pakistan, and Germany. And Mandiant also wrote that it was "aware of multiple cases where large volumes of files have been stolen from victims' MOVEit transfer systems" and adding that the web shell left behind, which they call LEMURLOOT, is also capable of stealing Azure Storage Blob information.

So we have, to start with, massive data exfiltration. And Mandiant's mention of a web shell brings us to the second of the two things that is being done, the implantation of new files and malware, because the bad guys have also been found to be dropping web shell malware before they leave. We've previously talked about web shells, but here's a bit of history. In the early days of the web, web servers only delivered static HTML web pages. You gave them a URL which was basically the location of the page's text on the server, and it returned that page to the browser.

After several years of that, browser-side scripting, which was embodied by and enabled by JavaScript, introduced the concept of running code that you received from a web server on your browser, so-called client-side scripting. This brought the user's client alive, giving it some of the capabilities of a local application. A perfect and simple example is that a user's entered password could be hashed locally by scripting running in the browser so that a web server that wanted a password never received anything but the hash. That was a big benefit in security for the user.

These days, of course, we're seeing the logical evolution of scripting on the client with amazingly complete and complex web apps. But even browser-side scripted pages could

be delivered by static files. You know, lots of JavaScript is just a .js, it's just a JavaScript file that is being delivered statically to drop a library of complex JavaScript like no.js onto the user's browser. That's just a static download, and typically it's cached, so it's very quick.

The big change on the server side occurred when web servers started running code to respond to queries being made by their clients. The earliest implementation of this was known as CGI, which stands for Common Gateway Interface. The idea behind CGI was simplicity. A web client would make a query, and the web server would essentially serve as an intermediary between the user's web browser and some code which the web server would execute on the server. To do this, the web server would launch and run a separate CGI program in the background, often with the extension .cgi.

The web server would provide the CGI program with what the user had queried, and whatever the CGI program returned through its Standard Output would be piped back by the web server to the user's web browser to be seen. So with CGI, rather than delivering static textual web pages, the output of a pre-compiled program was returned to the user's web browser. This was a big application for Larry Wall's PERL, which was often used in early CGI applications.

This model was clean and simple, and it remains in heavy use today where, as I noted earlier from that printers.php page, where PHP is the back-end recipient of a client's CGI queries. GRC's web forums and that link shortener I mentioned are all PHP, as is WordPress, which as we know runs a huge portion of the web, WordPress written in PHP. Modern web servers provide for many ways to generate dynamic content, as it's called. GRC's ShieldsUP!, Perfect Passwords, Perfect Paper Passwords, GRC's DNS Spoofability test, and GRC's support for SQRL are all implemented using DLLs that I wrote of course in assembly language, using the Microsoft ISAPI API to obtain and return data to and from users' web pages. So this is common.

Microsoft also promotes their own server-side interpreter which implements their scripting language known as Active Server Pages (ASP). When a web server which has Active Server Pages enabled encounters a URL referring to a file ending in .asp or .aspx, the web server will look for that file on the URL's provided path and will run the code contained in that file, the Active Server Pages code.

And that brings us back to the - I can't say it. Lemur, L-E-M-UR.

JASON: LEMURLOOT?

Steve: Lemur, thank you, lemur.

JASON: I don't know, I'm guessing.

Steve: If I start - I think you're right, it's LEMURLOOT.

JASON: LEMURLOOT.

Steve: LEMURLOOT web shell, a tongue twister.

JASON: Yeah.

Steve: Thank you, Jason. That's being left behind by these attackers. In machines that have been attacked, the attackers leave behind a file named "human2.aspx." They chose that name since a human.aspx file is already present in the system as part of the authentic file set. So obviously they're making it look like it belongs there.

They also sometimes leave additional files with the file extension ".cmdline." This human2.aspx file functions as a web shell. It's a sophisticated script that will provide future access to any ASP-script capable web server that's unlucky enough to host it. The bad guys know its name, and they know its location on the server. So they're able to invoke it at any time in the future remotely simply by querying the server for a URL and path ending in "human2.aspx." The web server will immediately run that code, which typically gives remote attackers control of the system and probably of the network. The web shell is also engineered to add new admin user account sessions with the name "Health Check Service" in an effort to appear benign and avoid detection.

This means that just patching against the attack after the fact will not be sufficient protection, since a previously vulnerable server may have already been quietly infected with the human2 web shell. After patching, it will be necessary to search for any "IOCs," as we call them today, Indications of Compromise.

CISA quickly issued a nationwide alert to let everyone know and to demand that all government agencies using the MOVEit Transfer system update and check for evidence of past incursions. Our friends at the web scanning search engine Censys, who were the subject of last week's podcast, have identified more than 3,000 vulnerable instances of MOVEit, the majority of which are located in the U.S.

Huntress Labs was all over this at the start of the month. Excerpting and editing a bit what they wrote, they said: "On June 1st, 2023, Huntress was made aware of active exploitation attempts against the MOVEit file transfer application. Previously, on May 31st, the vendor Progress had just released a security advisory expressing there is a critical vulnerability that could lead to unauthorized access. On June 2nd, the industry dubbed this vulnerability CVE-2023-34362. Progress brought down MOVEit Cloud as part of their response and investigation.

"Huntress has fully recreated the attack chain exploiting MOVEit Transfer software. We have uncovered that the initial phase of the attack, SQL injection, opens the door for even further compromise - specifically, arbitrary code execution. We use our exploit to receive shell access with Meterpreter, escalate to NT AUTHORITY\SYSTEM" - meaning full system-level privileges - "and detonate a Cl0p ransomware payload." And we'll be talking about Cl0p in a moment.

They said: "This means that any unauthenticated adversary could trigger an exploit that instantly deploys ransomware or performs any other malicious action. Malicious code would run under the MOVEit service account user 'moveitsvc,' which is in the local administrators group." Meaning, again, full admin privileges. "The attacker could disable antivirus protections, or achieve any other arbitrary code execution. Another demonstration showcased compromising the MOVEit Transfer API and application itself. With that alone, we upload, download, and potentially exfiltrate files as a threat actor would.

"The behavior that the industry observed, adding a human2.aspx web shell, is not necessary for attackers to compromise the MOVEit Transfer software. It's an option that this specific threat actor chose to deploy for persistence, but the attack vector offers the ability to detonate ransomware right away. Some have already publicly reported attackers pivoting to other file names," that is, using other than human2 because that's now become known. They end: "The recommended guidance is still to patch and enable logging. From our own testing, the patch does effectively thwart our recreated exploit." Okay. So beginning of June, patch released. Unfortunately, as they say, the horses had left the barn. Actually, they had left the barn, and they had traveled quite a distance from the barn.

Microsoft attributed the MOVEit Transfer zero-day attacks to Lace Tempest, a threat actor previously linked to Cl0p ransomware, data theft, and extortion attacks. On June 6th, the Cl0p gang posted a communication to their leak site demanding that their victims contact them before June 14th to negotiate extortion fees for deleting the stolen data. I'm going to read without correcting the grammar what the ransom note said that was posted on Cl0p's site for their victims to respond to.

Cl0p wrote: "Dear Companies. CL0P is one of top organization offer penetration testing service after the fact." Which, huh, offer penetration testing service after the fact. That's one way to put it. They said: "This is an announcement to educate companies who use Progress MOVEit product that chance is that we download a lot of your data as part of exceptional exploit. We are the only one who perform such attack, and relax because your data is safe." Yeah, right, in Russia. They said: "We are to proceed as follow, and you should pay attention to avoid extraordinary measures to impact your company. IMPORTANT!" In all caps with an exclamation point. "We do not wish to speak to media or researcher. Leave."

Then we have three steps. "Step 1: If you had MOVEit software, continue to Step 2, else leave. Step 2: Email our team at unlock@rsv-box.com or unlock@support-multi.com. Step 3: Our team will email you with dedicated chat URL over TOR. We have information on hundreds of companies, so our discussion will work very simple."

Now we have seven steps. "Step 1: If we do not hear from you until June 14th, we will post your name on this page. Step 2: If you receive chat URL, go there and introduce you. Step 3: Our test will provide 10% proof of data we have and price to delete. Step 4: You may ask for two to three files random as proof we are not lying. Step 5: You have three day to discuss price; and if no agreement, you custom page will be created. Step 6: After seven days all your data will start to be publication. Step 7: You chat will close after 10 not productive day, and data will be publish."

Next section is, all caps, "WHAT WARRANTY? Our team has been around for many years. We have not even one time not do as we promise. When we say data is delete it is cause we show video proof." Okay. I guess they're going to show someone typing the command into a console. Wow. That's proof, all right. It's definitely gone now. Then they say: "We have no use for few measle dollars to deceive you." They don't want any measle dollars, apparently. "Call today before your company name is publish here."

Friendly cop. "Friendly Cl0p," it says.

JASON: Cl0p, Cl0p, Cl0p.

Steve: "FRIENDLY CL0P. P.S. If you are a government, city, or police station, do not worry. We erased all your data. You do not need to contact us. We have no interest to expose such information."

Okay. Then, presumably after the initial June 14th contact deadline had passed, they added updates. So under updates they have "Bissell.com 50TB company data. Get ready for something interesting. Fmeraldc.com 100TB company data. Get ready for something interesting." And then a bit later, "360EquipmentFinance.com files, Part 1 Published. PrecisionMedicalBilling.net files, Part 1 Published. HCI.edu files, Part 1 Published."

So the point of this story is, over the past three weeks, is that we have the internal private data of thousands of U.S. companies who have been using Progress's MOVEit Transfer software, first exfiltrated from their servers, and in some cases as much as 100TB worth, apparently. This was not good, but we're not done yet. After a week had passed, believe it or not, Progress announced the news of additional discovered vulnerabilities.

June 9th they said: "In addition to the ongoing investigation into vulnerability CVE-2023-34362, we have partnered with cybersecurity experts to conduct further detailed code reviews" - oh, what a nice idea, let's do it now - "as an added layer of protection for our customers. As part of these code reviews, cybersecurity firm Huntress has helped us to uncover additional vulnerabilities that could potentially be used by a bad actor to stage an exploit. These newly discovered vulnerabilities are distinct from the previous reported vulnerability shared on May 31st, 2023. All MOVEit Transfer customers must apply the new patch, released on June 9th, 2023. All MOVEit Cloud customers, please see the MOVEit Cloud Knowledge Base Article for more information. The investigation is ongoing, but currently we have not seen indications that these newly discovered vulnerabilities have been exploited."

Then, exactly one week after that, last Friday, June 16th, we have more. "June 16th, 2023: Yesterday we reported the public posting of a new SQL injection vulnerability that required us to take down HTTPS traffic for MOVEit Cloud and to ask MOVEit Transfer customers to take down their HTTP and HTTPS traffic to safeguard their environments. We have now tested and deployed another patch to MOVEit Cloud, returning it to full service across all cloud clusters.

"We have also shared this patch and the necessary deployment steps with all MOVEit Transfer customers. All MOVEit Transfer customers must apply the new patch, released on June 16th, 2023. Details on steps to take can be found in the following Knowledge Base Article. All MOVEit Cloud customers, please see the MOVEit Cloud Status page for more information. The investigation is ongoing, but currently we have not seen indications that this newly discovered vulnerability has been exploited." So what happened in the third case is that some random independent person looked at the software and said, "Oh, how about this one?" and posted it publicly. Whoops.

Okay. So what do we know about the victims so far? CNN Business had some reporting on this. Excerpting from what CNN reported: "A growing number of businesses, universities, and government agencies have been targeted in a global cyberattack by Russian cybercriminals that are now working to understand how much data was compromised. CISA said Thursday that several federal agencies have experienced intrusions. The U.S. Department of Energy said it took immediate steps to mitigate the impact of the hack after learning that reports from two department entities had been compromised. It's also impacted state governments in Minnesota and Illinois. And on Thursday, state agencies said 3.5 million Oregonians with driver's licenses or state ID cards had been impacted by a breach, as well as anyone with that documentation in Louisiana." In other words, all of Louisiana.

"British Airways confirmed that its staffers' names, addresses, national insurance numbers, and banking details were exposed because its payroll provider Zellis used MOVEit. The BBC said its staff had also been afflicted because Zellis was its payroll provider. The UK's beauty and health company Boots said some of its team members' information was also stolen. Brett Callow, threat analyst at cybersecurity firm Emsisoft, said the hackers have also listed Aon and The Boston Globe as victims.

"'By my count,' he said, 'there are now 63 known/confirmed victims plus an unspecified number of USG agencies.' The hacking campaign has also spread to academia. Johns Hopkins University in Baltimore and the university's renowned health system said in a statement that 'sensitive personal and financial information,' including names, contact information, and health billing records, may have been stolen in the hack. Meanwhile, Georgia's statewide university system, which spans 40,000-student University of Georgia along with over a dozen other state colleges and universities, confirmed it was investigating the scope and severity of the hack."

TechCrunch added: "Cl0p, the ransomware gang responsible for exploiting a critical security vulnerability in a popular corporate file transfer tool, has begun listing victims of the mass hacks, including a number of U.S. banks and universities. The victim list, which was posted to Cl0p's dark web leak site, includes U.S.-based financial services organizations 1st Source and First National Bankers Bank, Boston-based investment management firm Putnam Investments, the Netherlands-based Landal GreenParks, and the U.K.-based energy giant Shell. GreenShield Canada, a non-profit benefits carrier that provides health and dental benefits, was listed on the leak site but has since been removed.

"Other victims listed include financial software provider Datasite, educational non-profit National Student Clearinghouse, student health insurance provider United Healthcare Student Resources, American manufacturer Leggett & Platt, Swiss insurance company OKK, and the University System of Georgia. A spokesperson for German mechanical engineering company Heidelberg, which Cl0p listed as a victim, told TechCrunch in a statement that the company is 'well aware of its mentioning on the Tor website of Cl0p and the incident connected to a supplier software.'

"Cl0p, which like other ransomware gangs typically contacts its victims to demand a ransom payment to decrypt or delete their stolen files, took the unusual step of not contacting the organizations it had hacked. Instead, a blackmail message posted on its dark web leak site told victims to contact the gang prior to its June 14th deadline. Multiple organizations have previously disclosed they were compromised as a result of the attacks, including the BBC, Aer Lingus and British Airways. These organizations were all affected because they rely on HR and payroll software supplier Zellis, which confirmed that its MOVEit system was compromised.

"The Government of Nova Scotia, which uses MOVEit to share files across departments, also confirmed it was affected, and said in a statement that some citizens' personal information may have been compromised. However, as we know, Clop's leak site said, 'If you are a government, city or police service, we erased all your data.' Ofcom, the U.K.'s communications regulator, also said some confidential information had been compromised in the MOVEit mass hack. In a statement, the regulator confirmed that hackers accessed some data about the companies it regulates, along with the personal information of 412 Ofcom employees. Transport for London (TfL), the government body responsible for running London's transport services, and global consultancy firm Ernst & Young are also impacted, according to BBC News. Neither organization responded to TechCrunch's questions."

Boy. You know, it's one thing to say, oh, thousands. It's another thing to actually hear them listed and understand what it means that that many actual physical entities were affected and afflicted by this. Since we haven't yet enumerated the literal thousands of individual companies, government and educational agencies, and other organizations that that have been compromised in this mass attack, many more victims are expected still to be revealed in the coming days and weeks.

And now that they know what to look for, security researchers looking back through their logs have determined that someone had been experimenting with the exploitation of these MOVEit vulnerabilities for the past two years, since 2021. And Cl0p was also responsible for previous mass attacks exploiting flaws in Fortra's GoAnywhere file transfer tool - remember that one recently? - and Accellion's file transfer application.

So, welcome to our new normal. A serious flaw is silently discovered in a popular, highly used, web-connected application. Its discoverer remains quiet for years while patiently working out exactly how to set up the attack for maximum effect in the greatest number of cases. There's a bit of a risk/reward tradeoff here, since it's always possible that by not jumping on and exploiting a vulnerability immediately, it'll give it some time to be

discovered and remediated before an attack can occur. We've often seen this effect when, for example, a well-planned Pwn2Own exploit fails because patches were coincidentally released on the eve of the competition to foreclose the exploit. Then, all vulnerable service instances are located ahead of time, and the attack is staged and readied.

And finally, effectively all at once, all of that vulnerable service's users have their data silently exfiltrated and stored. In the case of Cl0p, all of that sensitive data probably lands in Russia, and a web shell is also installed to allow future access. Finally, victim companies are notified, threatened en masse, publicly shamed, and eventually, if they don't accede to the extortionist's demands, have their potentially sensitive internal and client data released to the world.

The industry has observed that, in this instance, the traditional "don't call us, we'll call you" model has been reversed, with the attackers asking their victims to initiate the contact. It's been suggested, as I noted, by Tech Crunch that this is due to the fact that there are just too many victims for the attackers to manage proactively. So they have chosen to be more passive and wait to be contacted. This is probably an optimal strategy, since what the attackers want is maximum extortion payments. And the likelihood of being paid, and being paid a larger amount, is far higher if they are proactively contacted by a concerned victim than if they reach out to cajole all of their targets.

Now, at this point we don't know how much money the campaign will net for Cl0p. We're still in the early days, unfortunately. But the numbers are distressing for ransomware and cyber extortion gangs in general. Ryuk is known to have netted $150 million. REvil, $123 million in 2020. LockBit, $91 million. DarkSide, $90 million. Maze/Egregor, $75 million. Cuba, $44 million. Conti, $25.5 million. NetWalker, $25 million. Dharma, $24 million. All told, that comes to just shy of $650 million that has been raked up by these guys. And of course everybody knows this.

As this podcast has observed, malicious hacking is no longer being done for sport. It's now all about money. And unfortunately, money creates incentive. And as we've also seen frequently, there are a sufficient number of undiscovered vulnerabilities lurking within much of today's software to incentivize the bad guys into finding and exploiting them for their profit.

And it's not as if the bad guys are smarter. As soon as equally talented security researchers began taking a closer look at Progress's MOVEit Transfer software, additional previously unsuspected vulnerabilities started falling out of the thing weekly. That widely used software turned out to be a mess. Yet good guys were never given sufficient prior reason to examine it because the economics of doing so did not make sense. No security researcher was going to earn millions of dollars by discovering those problems and turning them in for a bounty. But the economics for the bad guys did and does make sense, since they will likely manage to extort millions of dollars overall from their newly acquired victim base.

Something needs to change. Academics in their ivory towers are busily inventing and developing new computer technologies that have none of these problems. But what we know is that down here on the ground, nothing changes unless it is forced to. When I was previously complaining about the utterly and obviously broken traditional model of SQL database access by web servers, I was scolded by our listeners and told, "Oh, Steve, don't you know that was the old way of using SQL?" Right? Old. Tell that to the thousands of victims of this latest catastrophe of SQL database usage.

And speaking of old, does everyone know just how old this attack is? Just how old is the exploit that created the "Little Bobby Drop Tables" joke? The operation of the SQL

injection exploit was first documented in 1998 by cybersecurity researcher and hacker Jeff Forristal. His findings were published in the hacker magazine Phrack. Writing under the moniker Rain Forest Puppy, Jeff explained how someone with basic coding skills could piggyback unauthorized SQL commands into legitimate SQL commands to pull sensitive information out of a website's database. Gee, doesn't that sound a lot like what just happened last week? And that warning came 23 years ago. 23 years ago. This is a fundamental database architecture that was horribly bad then, and nothing has changed since. It happened again three weeks ago.

In '98, when Jeff Forristal notified Microsoft about how the vulnerability impacted their very popular SQL Server product, Microsoft - no one's going to believe this. Microsoft didn't see it as a problem. As Forristal put it at the time in his article for Phrack: "According to Microsoft, what you're about to read is not a problem, so don't worry about doing anything to stop it." So how did that advice work out?

In 2007, the biggest convenience store chain in the United States at the time, 7-Eleven, fell victim to a SQL injection attack. The Russian hackers used SQL injections to hack into the 7-Eleven website and used that as a stepping stone into the convenience store's customer credit card database. This allowed the hackers to then withdraw cash back home in Russia. Wired magazine reported that the culprits absconded with $2 million. That same year, cybercriminals used SQL injection to gain administrative control over two U.S. Army-related websites and redirect visitors to websites with anti-American and anti-Israeli propaganda.

The next year, in 2008, MySpace data breach ranks as one of the largest attacks on a consumer website in history. Cybercriminals stole emails, names, and partial passwords of almost 360 million MySpace user accounts. Thanks to that attack, we learned that it wasn't a good idea to reuse passwords across sites.

The award for the most egregious lack of security probably goes to Equifax. The 2017 Equifax data breach which yielded extremely personal information - names, social security numbers, birthdays, and addresses - for 143 million consumers was, you guessed it, a SQL injection attack. And what's worse, prior to the data breach a cybersecurity researcher had warned Equifax that they were susceptible to a SQL injection attack. Whoops.

Every three years the OWASP Open Web Application Security Project ranks the Top 10 Most Critical Web Application Security Risks. Guess where SQL injection ranks? Yep, number one. Our listeners, who kindly took the time to educate me about there being much better and more secure ways to use SQL, were absolutely correct. Which only serves to underscore the tragedy of the fact that SQL will still happily operate today the same way it did 23 years ago, in 1998.

Using SQL the horribly and fundamentally insecure way is, unfortunately, also the obvious and easy way. This thing should have been strangled in its crib the moment it was born in 1998. But instead, Microsoft and others blessed it and said: "Oh, it's so wonderfully easy and powerful. Those worrywarts are just trying to get some ink. It's fine. Don't worry about it. Just be careful." Right. Be careful walking on ice while carrying that dynamite.

JASON: Just don't fall. Just, you know.

Steve: Just don't fall. And what was interesting was that notice that Huntress who found that second round of multiple exploits missed the third one. So even they, staring at the code, didn't see it. And that's the problem. This approach is insecure by default. And so it's up to someone attempting to filter out any possible misuse. And obviously that's not working out very well.

JASON: Mm-hmm. Mm-hmm. Wow. That is a lot to take in, and to know that there is no resolve, there is no resolution on anything, what you're talking about right now. It's worse. It just continues to snowball.

Steve: You know, and normally we're able to put some of the blame on the people who get attacked; right? It's like, well, you shouldn't have been doing this or that. You know, someone clicked a link, and that brought the malware in, and the ransomware got you. No. This was a widely used, you know, basically it's a bit of a competitor to Microsoft's file sharing solution, you know, SharePoint. It does the same sort of thing for enterprises. And so this thing was widely used. There was no reason to imagine that it would wipe out every one of its users.

JASON: Trust no one or nothing. Period. End of story. Because it all could go down.

Steve: I've got databases here, and none of them run SQL.

JASON: There you go.

Steve: My ecommerce system and SQRL need a user ID database on the backend. Nope, you won't find me using that piece of never mind.

JASON: Piece of never mind. I like that, actually. Good stuff. I love the deep dive. But, I mean, you always do fantastic deep dives. But I often find myself at the end of it like, oh, dear god. Like, what are you going to do?

Steve: Well, Jason, now we've got the power LEDs on all of our security equipment.

JASON: I know. If it's got a little LED power on it, then it's over. We're screwed. It's like, even that isn't safe. If power LEDs aren't safe, what is? That's what I ask you. Steve, great stuff. Anyone who wants to get into everything that Steve's up to, all you have to do is go to GRC.com. You'll find all sorts of Steve Gibson goodness there, SpinRite of course. Again, congratulations, major milestone.

Steve: Thank you.

JASON: And if you aren't onboard, you can get it there. The best mass storage recovery and maintenance tool can be found at GRC.com. Audio and video of this show, of course. Also transcripts of this show can be found there, as well. If you are on the TWiT site, you want to find Security Now!, it's easy to do: TWiT.tv/sn. Of course we have our audio and video there, as well. But we also have ways to subscribe to the show, which is really at the end of the day one of the most important things. If you love what we're doing here at TWiT, if you love podcasting in general, subscribe to the shows. That's sending the best signal to us and to the people who help pay us to do what we do. So that can be found at TWiT.tv/sn.

We do record this show every Tuesday. Usually it's Leo sitting here in the studio. But if not, I'm happy to fill in for Leo any time he's out at Disneyland. But we do it every Tuesday, 4:30 p.m. Eastern, 1:30 p.m. Pacific, 20:30 UTC. Thank you so much to John for engineering here in the studio. Thank you to Steve. You do a fantastic show, and I'm always honored to sit in the Leo hot seat while he's out. So thank you for welcoming me, Steve.

Steve: Jason, absolute pleasure, and I'll see you next time you're standing in.

JASON: Sounds good. We'll see you next time on Security Now!. Bye, everybody.

Steve: Bye.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details: http://creativecommons.org/licenses/by-nc-sa/2.5/