SeriousSAM & PetitPotam

Description: This week we will plow into another two new serious vulnerabilities brought to the industry by Microsoft named SeriousSAM and PetitPotam. But we first look at how Chrome managed to hugely speed up its phishing website early warning system, making it even earlier. We cover the striking news of Kaseya having obtained a universal decryptor which is effective for every one of their victims. We look at the massive HP printer driver mess and consider the larger lesson that it teaches. We look at the new security features GitHub is bringing to its support of the "Go" language. Then, after sharing one bit of listener feedback, we plow into SeriousSAM and PetitPotam.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-829.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-829-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with a potpourri of great, interesting tech news including how Chrome is making it easier to detect phishing sites using the colors on the page. Kind of a clever fingerprinting algorithm. We'll talk about a printer driver used by millions of HP, Samsung, and Xerox printers that is highly vulnerable. And then we'll detail a couple of new Windows exploits. And Steve will explain why he's no fan of how Microsoft does things these days. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 829, recorded Tuesday, July 27th, 2021: SeriousSAM & PetitPotam.

It's time for Security Now!, the show where we cover your security, your privacy, online and off, with this guy right here, the chief, the man in charge, the explainer in chief, Steve Gibson. Hi, Steve.

Steve Gibson: Yo, Leo. Great to be with you once again. We've got a bunch of stuff to talk about. I feel so strongly about some of these topics that I thought maybe we should call this podcast Security With Attitude because we're going to be getting some this hour. Okay. This is Episode 829 for - this is our last podcast of July. Somehow we've survived July. Well, okay, wait. We have four more days, so we'll hope that we're going to survive.

We're going to talk about two things. We're going to plow into another two new serious vulnerabilities brought to the industry by Microsoft named SeriousSAM & PetitPotam. And, you know, SeriousSAM, it was happening, it was breaking just as we were recording last week, referred to also as the Hive Nightmare. But that was because of the printer nightmares that we've been surviving and talking about so far during the month. So the person who came up with that, you know, it first got named that. But SeriousSAM is better. And of course it rhymes with PetitPotam.

Leo: Did you ever play SeriousSAM the game?

Steve: No. As a matter of fact, when I googled to do some in-depth looking, I realized, oh. And I was a little sucked into the Wikipedia entry. It was sort of fascinating how back in the year 2000, it was about 20 years ago, the game engines cost a million dollars.

SAM: Hey, guys, let's play...

Leo: That was SeriousSAM, just so you know.

SAM: I won't pass by. Let's fight.

Leo: Okay, just so you know, one of the great classic games of all time.

Steve: I guess. So I was unaware, but now I know.

SAM: Just what the hell was that, anyway?

Leo: All right. Thanks, Sam. You're done.

Steve: Okay. So we're going to look at how Chrome has managed to hugely speed up its phishing website early warning system, making it even earlier. We cover the striking news of Kaseya having obtained a universal decryptor which is effective for every one of their victims.

Leo: I wonder where they got that.

Steve: Uh-huh. We look at the massive HP printer driver mess and consider the larger lesson it teaches. And then we look at the new security features GitHub has just announced that it's bringing to its support for the Go Language. And after sharing one little bit of listener feedback, something I've mentioned before, just a useful reminder, we're going to plow into SeriousSAM and PetitPotam.

Leo: These are very creative names, I know.

Steve: And of course we have a great techie-oriented Picture of the Week that has been in my queue for a while, ever since I saw it. And I thought, okay, now's a good time to share it with our, well, our viewers. And I can describe it to our listeners. So I think another great podcast for our...

Leo: Thank you for considering both. I do like it that you do that. You're very good at describing those. Picture of the Week, Steve?

Steve: Yeah. So this one, it's fun.

Leo: Oh, it's hysterical.

Steve: At first blush you'd think, okay, Photoshop, because it looks like it was photoshopped. This was the conclusion of a stunning illuminated drone show. This is 1,500 drones flying around in the air that assembled at the end of the presentation to present the QR code, which is valid and scannable, to the URL of the game company that put this on.

Leo: Wow.

Steve: First it showed animated characters from the game, like swords being swung and all kinds of crazy stuff. And frankly, I'm in awe of the idea that somehow they've developed the technology to microposition drones relative to each other with this kind of precision and animate it. Anyway, it had been in the queue of things to show. I just thought it was so cool that I wanted to share it with our listeners. This took place in Shanghai. I do have a link in the show notes to the article at Vice.com which talked about it. And if anyone was curious, I did see the animated videos, and it's just astonishing.

Leo: We were talking about this actually in our TWiT forums, twit.community forums. And I found - because somebody said, well, how do they do that? And I found a really good article at ScienceDirect.com that describes in great detail how these drone swarms, is what they call them, are positioned.

Steve: Are positioned so accurately.

Leo: It's fascinating. And it's multilayered. They all have cameras. But Intel's famous, they've got the drone - we could do it, Steve. Just $90,000. They've got, I don't know what it is, a couple thousand drones that will - we saw it at the Olympics a couple years ago; at the Super Bowl. It's really, really, really cool. So there's a whole article.

Steve: I think Biden's inauguration or his something...

Leo: Yeah.

Steve: I think there was something in addition to fireworks, as I recall, also, that they were showing.

Leo: Yeah. Well, it's better than fireworks nowadays because it's not so damaging.

Steve: Yeah.

Leo: They're reusable, obviously. But it's a really incredible thing because these things fly within inches of each other, perfect formation.

Steve: Exactly. And they've got high-speed spinning props.

Leo: Yeah.

Steve: It's not like nothing. Yeah, it's just very, very, very cool.

Leo: Intel kind of invented a drone specifically to do this. Yeah. Whoops, that's my salad. If I could have an Intel caprese drone, I'd be very interested. But look what they can do. I mean, it's just amazing. It's really cool.

Steve: That's drones?

Leo: Yeah. That owl? Yeah. Look at this.

Steve: Oh, very - oh.

Leo: Much better, I think, than fireworks.

Steve: Yeah, yeah, yeah. Very cool.

Leo: All right. Back to - I'm sorry. I didn't mean to get sidetracked.

Steve: Let's see. What were we - why are we here?

Leo: What are we doing? What are we talking about today?

Steve: Okay. Actually, we do have a lot to talk about. So I'm reading this Chromium blog posting titled "Faster and More Efficient Phishing Detection in M92," which is the just-released version of Chrome. You know, because that sounds like a good thing. And depending upon what the posting's details revealed, I figured that it might be of interest to our listeners. You're hearing about it, of course, because that didn't turn out to be the case, though perhaps not for the reason you might imagine.

The posting starts out with a little introductory marketing spiel. They said: "Keeping Chrome users safe as they browse the web is crucially important to Chrome; in fact, security has always been one of our four corner principles." Then I of course was wondering what the other three were, but anyway. "In some cases, security can come at the expense of performance. In this case" - and of course they named their series "The Fast and the Curious."

Leo: Oh, god.

Steve: I know. "We are excited to share how improvements to our phishing detection algorithms keeps users safe online. With these improvements, phishing detection is now 50" - five oh, and it turns out it's actually a statistical spread, but we'll get there in a second - "50 times faster" - it should say "as much as" - "50 times faster and drains less battery."

Then under the subheading of "Phishing Detection" they write, and they begin to explain this, and this is what I had to like do a double-take: "Every time you navigate to a new page, Chrome evaluates a collection of signals about the page to see if it matches those of phishing sites." Okay. "To do that, we compare the color profile of the visited page."

Leo: What?

Steve: I know. "That's the range and frequency of colors present on the page."

Leo: That's - what does that have to do with phishing?

Steve: Exactly, it's nuts, "with the color profiles of common pages. For example, in the image below, we can see that the colors are mostly orange, followed by green and then a touch of purple." Now, in the show notes, I have that captured. Of course those listening cannot see the diagram which I've included in the show notes. But it's a page with a bunch of orange pumpkins so that orange dominates the page, though they are all in light blue frames. It actually has more surface area. For some reason in their example they're ignoring the light blue. Okay. Maybe they actually do ignore the background. I don't know. Actually, later on it looks like they don't. But they also pick up on one of the frames which has a green background. But in any event, this says that to detect phishing they're looking at a page's color distribution. So my first thought was like yours, Leo. What?

Leo: What? Why? Yeah.

Steve: They're comparing the color profile of a page we visit to the color profiles of common pages? Really? Turns out yes, they are. That's what Chrome does. Then it hit me. Whatever they do to pull off this detection needs to be done entirely on the client.

Leo: Right.

Steve: Right? Chrome cannot be sending all visited page URLs back to the Google mothership. That would be a privacy catastrophe. Okay, now, and we'll ignore for the time being that loading our web pages down full of image beacon pixels and JavaScript is essentially doing exactly that. But okay. Not formally. And then they confirm the nature of the strategy by explaining: "If the site matches a known phishing site, Chrome warns you to protect your personal information and prevent you from exposing your credentials. To preserve your privacy, by default Chrome's Safe Browsing mode never sends any images outside the browser. While this is great for privacy, it means that your machine has to do all the work to analyze the image."

Okay. So then what follows I've lightly edited to clarify what they're saying without the use of any graphics. So they essentially wrote: "Image processing can generate heavy workloads because analyzing the image requires an evaluation of each pixel in what is commonly known as a 'pixel loop.' Some modern monitors display upwards of 14 million pixels, so even simple operations on each of those pixels can add up to a lot of CPU use. For phishing detection, the operation that takes place on each pixel is the counting of its basic colors."

Leo: Wow.

Steve: Yeah. This was all news to me.

Leo: Did they explain why they're monitoring the colors? I'm sure there's some correlation, but I just - it's not obvious.

Steve: So they say, yeah, kind of. "The colors are stored in an associative data structure called a hash map. For each pixel, we extract its RGB color values and store the counts in one of three different hash maps." So I guess they break it down into RGB, and then they're storing a histogram of the individual intensities of each R, G, and B component.

Leo: Yeah, they probably hash it for speed; right?

Steve: And then hashing. Right. And they said: "One for each color. Adding one item to a hash map is fast, but we have to do this for millions of pixels. We try to avoid reducing the number of pixels to avoid compromising the quality of the analysis. However, the computation itself can be improved; and in this just-released 92, it has been. The code now avoids keeping track of RGB channels in three different hash maps and instead uses only one to index by color. Three times less counting," they said. "And" - this I think is the big key - "consecutive pixels are summed before being counted in the hash map."

Leo: It's kind of like Huffman encoding.

Steve: Yeah, right. "For a site with a uniform background color" - thus the reason I said, well, they actually are paying attention to background colors - they said, "this can reduce the hash map overhead to almost nothing." Like in the same way as you said, Huffman would compress a long run of something down to something very short. "With the new approach, there are significantly fewer operations on the hash map. As a result, starting with 92, Chrome now executes image-based phishing classification" - okay, so the big takeaway is that phishing is using image-based classification, like, okay, who knew that? - "up to 50 times faster at the 50th percentile."

Okay, so that means what, that half of the users get 50 times improvement. On one side of the 50th percentile they're getting even better improvement, and on the other side they're getting less dramatic improvement. But they did say "and 2.5 times faster at the 99th percentile." So even 99% of all users will get up to at least 2.5 times faster. "On average," they said, "users will get their phishing classification results after 100 milliseconds, instead of 1.8 seconds." They said: "This benefits you in two ways as you use Chrome. First and foremost, using less CPU time to achieve the same work improves

general performance. Less CPU time means less battery drain and less time with spinning fans.

"Second, getting the results faster means Chrome can warn you sooner. The optimization brought the percentage of requests that took more than five seconds to process from 16.25% to less than 1.6%." So 16.5% were actually requiring more than 5.5 seconds for Chrome to decide if this was a phishing page you were being shown. So this client-side phishing detection has been very time and compute intensive. And they said: "The speed improvement makes a real difference in security, especially when it comes to stopping you from entering your password in a phishing site." Yeah, five seconds.

And so if you've got any kind of automated username and password insertion going on, you may have already done that. It will have filled in the form, and you've clicked "log me in" before five seconds have passed. And then it turns out to be a malignant, malicious site. So yeah, you don't want to wait, I mean, you want to know a lot faster than that.

They said: "Overall, these changes achieve a reduction of almost 1.2% of the total CPU time used by all Chrome renderer processes and utility processes. At Chrome's scale, even minor algorithm improvements can result in major energy efficiency gains in aggregate. Here's to many, many more centuries of CPU time saved. Stay tuned for many more performance improvements to come." So what they must have done is done some performance profiling of Chrome and looked at, like, where all the time was going, and a chunk of it was going to looking at every single page the user visits. And is it valid?

And so this also means that in every end-user's Chrome repository on their local machine are a set of criteria that, like, phishing sites they have seen and profiled. So they've done this wacky image distillation and basically reduced the image to a hash, and they've stored all these hashes, lord knows how many, on our hard drives. And when you go to a page, Chrome performs this process and checks to see if there's a match with any known phishing images and, if so, warns you.

Leo: It's essentially fingerprinting the site, but it's doing so by its use of color. And I bet you where this came from is you have a similar problem if you're trying to detect porn. And so all of the companies, Facebook, Google, anybody who's got image storage has an algorithm, especially not just porn in general, but revenge porn, where they have hashes of known revenge porn images, and then they look for...

Steve: Any future occurrence.

Leo: If it matches, oh, we've got another one, and they pull it down immediately. And I suspect - and they also, I think there were attempts to do this with just too much flesh at one point, which by the way were not very effective. But that's probably where the body of knowledge comes from.

Steve: Oh, so it used to be heuristic.

Leo: Heuristic, yeah.

Steve: So if it were a lot of flesh tones, eh.

Leo: A lot of flesh tones, eh. That didn't work so well. I remember they stopped doing that. But I suspect that the research done is probably related to this. So what's interesting is the choice and use of colors in a site are unique enough that you can say it's a fingerprint, essentially. Very interesting.

Steve: Yup. And the other nice thing about this is, when you think about it - because we're all twitchy now about fingerprinting; right? So this is something that absolutely destroys the image. Like you're not in any way storing a representation of an image that could be sensitive. It's a hash of some deconstruction of the individual RGB values which are counted in some fashion and just, you know. So, I mean, it's really - and the fact that they're using a hash means that it can't be reversed; right?

Leo: That was the image with revenge porn. You don't want to circulate those images.

Steve: Right.

Leo: That would be counterproductive.

Steve: And as we know, anytime you pump something through a hash, that's an information lossy process, in this case on purpose. So they've come up with something which is certainly not inexpensive to do, and it's like, wow, I didn't realize that was happening in the background. And I've had that big, you know, I'm sure we've all, Chrome users, the whole screen goes red.

Leo: Oh, yeah.

Steve: And it's like, "Warning, Will Robinson." It's like, whoa, what happened? There it is, yup, there is a perfect sample of it that you just brought up.

Leo: Deceptive site ahead.

Steve: Right.

Leo: I just think it's a - this is a very kind of computer science-y solution. Because the problem is, well, how do you, okay, we've got known phishing sites. How do you fingerprint those and identify them in a speedy way in future? And I love it that they said, you know, if we just look at the colors, that's enough.

Steve: Yeah.

Leo: I think it's great. And we can hash those very quickly.

Steve: And so the takeaway is probably do not put too many pictures of pumpkins on your...

Leo: Stay away the pumpkins.

Steve: Might cause a false positive. Because pumpkins could be misconstrued.

Leo: You never know. You never know.

Steve: Okay. So this was very cool in our ransom news section. A universal decryptor for all Kaseya victims. The industry recently received some news from Kaseya, who recently posted a pair of interesting updates. The first one was posted last Thursday, on July 22nd, at 3:30 p.m. Eastern. This was their announcement.

They said: "Kaseya has obtained a universal decryptor key. On 7/21/2021" - so that was the day before their posting, so last Wednesday - they wrote, "Kaseya obtained a decryptor for victims of the REvil ransomware attack, and we're working to remediate customers impacted by the incident. We can confirm that Kaseya obtained the tool from a third party and have teams actively helping customers affected by the ransomware to restore their environments, with no reports of any problem or issues associated with the decryptor. Kaseya is working with Emsisoft to support our customer engagement efforts, and Emsisoft has confirmed the key is effective at unlocking victims.

"We remain committed to ensuring the highest levels of safety for our customers and will continue to update here as more details become available. Customers who have been impacted by the ransomware will be contacted by Kaseya representatives."

Leo: Oh, by the way, and forced to sign an NDA.

Steve: Ah, interesting.

Leo: So, yeah.

Steve: I did see reference to the NDA, but I didn't track that down. Then yesterday, on Monday - oh, to sign an NDA not to disclose that they were a victim and, like, raise the profile of the attacks on Kaseya.

Leo: And I wonder if also Kaseya wants to hide the fact that they almost certainly bought this decryptor from the REvil gang; right?

Steve: Well, so then, yesterday, Monday, July 26th, at 1:00 p.m. Eastern, they updated what they had posted, writing: "Throughout this past weekend, Kaseya's Incident Response team and Emsisoft partners continued their work assisting our customers and others with the restoration of their encrypted data. We continue to provide the decryptor to customers that request it, and we encourage all our customers whose data may have been encrypted during the attack to reach out to your contacts at Kaseya. The decryption

tool has proven 100% effective at decrypting files that were fully encrypted in the attack."

Next paragraph: "Kaseya has maintained our focus on assisting our customers. And when Kaseya obtained the decryptor last week, we moved as quickly as possible to safely use the decryptor to help our customers recover their encrypted data. Recent reports have suggested that our continued silence on whether Kaseya paid the ransom may encourage additional ransomware attacks, but nothing could be further from our goal. While each company must make its own decision on whether to pay the ransom, Kaseya decided after consultation with experts to not negotiate with the criminals who perpetrated this attack, and we have not wavered from that commitment. As such, we are confirming in no uncertain terms that Kaseya did not pay a ransom, either directly or indirectly through a third party, to obtain the decryptor."

Leo: Oh. So where did they get it?

Steve: So thanks to our podcast, which unfortunately, Leo, you missed because it's what I did two weeks ago, that was titled "REvil's Clever Crypto," we know exactly how a single campaign-wide multi-system universal decrypting tool could be provided. Thanks to the dual side-by-side encryption of the private half of each system-specific key pair, either the REvil affiliate themselves could have been induced to provide their private key for the entire Kaseya campaign or, up a level in the hierarchy, the REvil gang themselves could have provided a universal decryptor which would have been effective for all of that affiliate's victims, among which Kaseya would certainly have been the most prominent.

Leo: As I remember, REvil was offering it. I think it was $20 million for the universal...

Steve: 70.

Leo: 70. Well.

Steve: 70. And so what we presume sort of that goes, I think corresponds to last week's podcast, which was titled "REvil Vanishes," we could presume...

Leo: That they got the money.

Steve: ...pressure. Well, either they got the money, or they got pressure; right? It's got to be the case that law enforcement in Russia, responding probably to political heat...

Leo: Maybe. Maybe you want to hand over key.

Steve: Yeah.

Leo: Would be good if you do.

Steve: And what's so cool about the nature of the hierarchy is that either the affiliate who may have also been knowable could have been told, okay, you're giving up this key for the Kaseya attacks because you have to. Or REvil could have been asked to provide the key to decrypt all of the affiliates' work. So we'll never know what transpired. But the good news is all of the Kaseya victims get decrypted. And I think it supports the contention, further supports it, that I voiced last week, that we can expect to see future ransomware attacks and attackers working to deliberately remain under the radar.

From the standpoint of REvil and their affiliate, the Colonial Pipeline, the JBS Foods, and the Kaseya attacks have been catastrophic for them. You know, they were not good results because they became far too public, and thus they became political. So anyway, I'm sure that the lesson that's been taken away, that they have taken away, is to make much smaller waves in the future. Yes, we know that there's money to be made, but not when you make too much at once. That's just not - that just doesn't have a good outcome here.

Okay. The printer driver used by, it's estimated, hundreds of millions of HP, Samsung, and Xerox printers turns out to be exploitable. A researcher by the name of Asaf Amir with SentinelLabs gave HP, Samsung, and Xerox, although primarily HP because, as we'll see, it was OEMed to the other two - a generous five-month vulnerability pre-disclosure quiet period of what he and his team discovered. So HP was notified last February, five months ago. Microsoft has incorporated its update into one of the very recent Windows updates. The SentinelLabs vulnerability disclosure was just published last Tuesday. It contains four bullet points by way of its Executive Summary.

They said: "SentinelLabs has discovered a high-severity flaw in HP, Samsung, and Xerox printer drivers. Since 2005, HP, Samsung, and Xerox have released millions of printers worldwide with the vulnerable driver. SentinelLabs' findings were proactively reported to HP on Feb 18th, 2021 and are tracked as CVE-2021-3438, marked with a CVSS Score of 8.8. HP released a security update on May 19th to its customers to address the vulnerability." And somewhere I saw that Microsoft had incorporated that update into their own Windows Update, thank goodness, otherwise this thing would be a serious pain.

So what Asaf and his team discovered was a trivial-to-exploit flaw affecting the printer drivers used by a large family of printers which have been continuously shipping since 2005. Leo, 2005 is the year we began this podcast.

Leo: Yeah.

Steve: So as long as this podcast has been running, all of the printer drivers in HP's printers have been shipping with this flaw.

Leo: Ugh.

Steve: Yeah. Here's the way Asaf describes what happened and what they found: "Several months ago," he wrote, "while configuring a brand new HP printer, our team came across an old printer driver from 2005 called SSPORT.SYS, thanks to an alert by Process Hacker once again." Now, we'll be looping back, and I'm going to tell everybody about Process Hacker so you don't need to take a note yet. He said: "This led to the discovery of a high-severity vulnerability in HP, Xerox, and Samsung printer driver software that has remained hidden for 16 years. This vulnerability affects a very long list

of over 380 different HP and Samsung printer models, as well as at least a dozen different Xerox products."

So the disclosure page shows a sample. These guys, SentinelLabs' vulnerability disclosure, lists some printers. But Leo, go to the page here in the show notes, at the top of page 5 of the show notes. And then expand under the affected products. It just says "affected products" with a little plus sign. Stand back when you click on that and start scrolling because this thing goes on and on and on and on and on.

Leo: Wow.

Steve: So, yeah. This is a list.

Leo: A lot of these are Samsung.

Steve: Yeah, well, because they were OEMing it from HP.

Leo: Oh, boy.

Steve: Yeah. "So just by running the printer software, the driver gets installed and activated on the machine, regardless of whether," he writes, "you complete the installation or cancel." You can even cancel when it starts to install. It doesn't matter. It already got in. "Thus, in effect," he writes, "this driver gets installed and loaded without even asking or notifying the user. Whether you are configuring the printer to work wirelessly or via a USB cable, this driver gets loaded. In addition, it will be loaded by Windows on every subsequent boot. This makes the driver a perfect candidate to target since it will always be loaded on the machine, even if there is no printer connected.

"The vulnerable function inside the driver accepts data sent from User Mode via IOCTL (Input/Output Control) without validating the size parameter. This function copies a string from the user input using strncpy" - standard library, C library function (S-T-R-N-C-P-Y) - "with a size parameter that is controlled by the user. Essentially, this allows attackers to overrun the buffer used by the driver and run code that they have provided."

Okay. So just stop for a minute. We have, for the last 16 years, every instance of those 380 HP printers and all of those Samsungs and a handful of Xeroxes, when the drivers touch your system, installs this SSPORT.SYS device driver, setting it up to run at boot. It contains an IOCTL API call that I'll explain in a second, which allows any program in user mode to gain system privileges on the kernel. And it's been in essentially all Windows systems that have any of these, that have ever touched any of these 380 different HP printers for the last 16 years.

Okay. Windows IOCTL API is explicitly a means for allowing unprivileged user mode code running in Ring 3 to communicate with drivers, with services and other kernel code in Ring 0. It's an officially sanctioned and supported Windows API. So what we have here once again is a trivial-to-exploit means for bypassing Windows' entire process and user security privilege model. Last week we were introduced to the idea of signing a malicious driver, a printer driver, for installation somewhere within an enterprise's network, whereupon Windows would auto load it through their PointAndPrint facility, which Microsoft subsequently assured us was "vulnerable by design," their exact words. Therefore, it truly wasn't a bug. It was a feature.

Now we learn that for 16 years, from 2005 until a month or two ago, when HP posted this on their site, which no one would see, but Microsoft presumably incorporated it into a Windows Update - maybe it was three weeks ago. Until a month ago, when this was fixed, for hundreds of millions of Windows systems worldwide it was no longer necessary to bother with even malicious printer drivers because a popular, often-installed printer driver would, you know, you didn't have to bother with all that muss and fuss. Just find any machine with the SSPORT.SYS device driver present, and pass a specially crafted IOCTL API call to it, immediately taking over the system.

And then they continued. The guys at Sentinel said: "An interesting thing we noticed while investigating this driver is the peculiar hardcoded string: 'This String is from Device Driver@@@@.'" They wrote: "It seems that HP didn't develop this driver, but copied it from a project in Windows Driver Samples by Microsoft that has almost identical functionality." They said: "Fortunately, the MS sample project does not contain the vulnerability." So they took the sample code; you know? Where have we heard this before, Leo? Remember UPnP, where Intel posted sample UPnP source, saying here's some code. Don't use it. And then of course everybody did. And it contained a flaw that then, surprise, everybody had.

Okay. So in this case the modification they made to the sample project was what induced the flaw, and that's what they've been shipping for 16 years. So they said, the Sentinel guys, the SentinelLabs guys: "An exploitable kernel driver vulnerability can lead an unprivileged user to a SYSTEM account and run code in kernel mode, since the vulnerable driver is locally available to anyone. Among the obvious abuses of such vulnerabilities are that they could be used to bypass security products. Successfully exploiting a driver vulnerability might allow attackers to potentially install programs; view, change, encrypt or delete data; or create new accounts with full user rights. Weaponizing this vulnerability might require chaining other bugs as we didn't find a way to weaponize it by itself given the time invested." In other words, they didn't bother taking the time. Bad guys would.

And they finished: "Generally speaking, it is highly recommended that in order to reduce the attack surface provided by device drivers with exposed IOCTL handlers, developers should enforce strong ACLs (Access Control Lists) when creating kernel device objects, verify user input" - uh-huh, that is, not give the user control over the length of the buffer that's allocated, which it then fills - "and not expose a generic interface to kernel mode operations."

Okay. Now, stepping back from the specifics, as I mentioned above, the world has apparently dodged another bullet. We don't know of this vulnerability having been discovered and used to perform effortless elevation of privilege attacks on Windows systems during the previous 16 years. But the ability to do so has been there since, as I said, the start of this podcast. But we should take note that this highlights a fundamental and significant security weakness in the architecture of Windows, which can never be remedied. We saw a strong hint of this design flaw last week, as I mentioned, when Mimikatz's Benjamin Delpy demonstrated how a malicious printer driver, once signed and installed into a low-value machine, could be proactively pulled throughout an enterprise by Windows PointAndPrint feature, which was designed to silently and automatically install any needed drivers so that its users didn't need to be bothered with any of that.

But Windows' problem is actually much worse, as this serious problem with a widespread HP printer driver has just demonstrated. Windows drivers, like core services, run in the kernel. Userland applications can communicate with them in two ways - through the normal data channel, which is opening a device and then sending output to it, you know, we might call this use "in band use" of the driver. But drivers can also be communicated with out of band through the use of the IOCTL API. A printer driver, for example, might provide IOCTL services to user mode applications for setting its page orientation,

checking on the toner level, or communicating readiness and error conditions. You know, all those things now that drivers do that's not just dumping the page out. That's all accomplished through the IOCTL API. So it exists. And it's heavily used.

So Windows provides both for in-band and out-of-band communication between applications and drivers. That's not unusual. But these drivers, as I mentioned, run with kernel-level privilege. And the out-of-band IOCTL API deliberately crosses privilege boundaries. This means that any flaw existing in any driver can be used by any code running on the system to compromise that system. And as we've just seen, these drivers are not all written, vetted, and provided by Microsoft. More often, they're provided by third parties to make their devices work with Windows. Therefore, any mistake made by any device driver vendor which is discovered by anyone malicious can potentially be used to compromise an entire system containing that driver.

A long time ago, in a galaxy far, far away, long before network security was even a thing, this design made sense because remote attackers operating from hostile foreign countries were not able to lurk in our machines. But we've often used the apropos model of a chain of individual links, where the strength of the entire chain is limited by the strength of its weakest link. In Windows, the weakest links may be flaws lurking in the services and device drivers provided by well-meaning developers whose code has never been thoroughly stress-tested and security vetted because the moment it started to work without crashing, it was declared finished. And it was packaged up and shipped.

So in that world, which is now unfortunately this world, it doesn't take much imagination to picture an attacker who gets into a machine as a low-privilege user. They take an inventory of the system's third-party services and device drivers, then cross-reference that inventory against their own internal stash of never disclosed, privately known, third-party service and device driver exploits to determine which entry point into the kernel this system will offer.

Leo: And the good news is, thanks to hashing, this now takes seconds less. It's probably pretty instantaneous.

Steve: Exactly. So this is a problem. We don't want to talk about it because there's nothing we can do about it. Our userland code has to talk through to the Ring 0 to talk to the device drivers that are down there with those privileges. This just happened to come to light when these guys thought, wait a minute, SSPORT.SYS. That's from 2005, 16 years ago. Has anyone looked at its security? And no.

Leo: No.

Steve: And what they found was a trivial-to-exploit buffer overflow. Now, this would be a disaster if this was something that Microsoft weren't able to fix quickly. The problem is, and this is what I hope I've - the point I've driven home. This is like Colonial Pipeline; right? It was so big that it got, you know, it was a disaster, and it got remedied. Or Kaseya. But what the ransomware guys now know they have to do is distribute their attacks. Unfortunately, what we have is a case of distributed drivers. Rather than, you know, this one, this big HP one was such a mess that Microsoft said, oh, crap, and like immediately gave it a Patch Tuesday fix. The problem is who knows what serious security flaws are lurking in all the little smaller third-party things that exist in Windows that nobody has bothered to take a look at. Again, I would not be surprised.

We talked about some time ago the model of vulnerability becoming publicly known for some service that has a public exposure on the Internet, and the bad guys have - no doubt, Leo, it's been hashed - a big hash table of port numbers that immediately lead them to the exploits they know about to take advantage of that, the appearance of that thing on some port. And they jump on it before it can get patched. So anyway, nothing can be done. But we are on essentially an increasingly brittle-seeming operating system platform. And as we're going to learn by the end of this podcast, thank you, as we're going to learn by the end of this podcast, even 11 doesn't fix it.

And before we take our second break, I wanted to mention, bring to our listeners' notice something that these guys used that brought this to their attention called Process Hacker. The SentinelLabs guys discovered this whole HP printer driver mess when this thing called Process Hacker proactively popped up a notification that this SSPORT.SYS service had just been created as a result of something they were doing. I for one would love the idea of being proactively notified when something has just added a background service or driver to my system. Maybe it's something I'm expecting. But if it's not, I want to know.

So I wanted to take a moment to shine a light on the tool they used known as Process Hacker. Many of us are familiar with Mark Russinovich's excellent Sysinternal Tools. One of them, Process Explorer, is immediately reminiscent of Process Hacker. It looks like - Process Hacker I guess I would describe as Process Explorer on steroids. And it's open source.

Leo: It's open source, that's awesome.

Steve: Open source. So Process Hacker has taken what Mark has done much further. It's open source. It's still at SourceForge, but it also has a GitHub presence. It runs on anything from Win7 on. It shows a download count of 6.5 million. It shows 34 contributors, 814 forks, and is being actively developed and maintained. It bills itself as a free, powerful, multipurpose tool that helps you monitor system resources, debug software, and detect malware. Its bullet-pointed feature list says a detailed overview of system activity with highlighting. Graphs and statistics allow you to quickly track down resource hogs and runaway processes. Can't edit or delete a file? Discover which processes are using that file. See what programs have active network connections and close them if necessary. Get real-time information on disk accesses and who's doing them. View detailed stack traces with kernel mode, WOW64, and .NET support. Go beyond services .msc; create, edit, and control services. Small, portable, no installation required. 100% free open software under GPL v3.

Leo: That's really great.

Steve: And it offers a plug-in architecture for extensions. So it's able to sit quietly in the background, alert when something is setting up permanent residence on any Windows machine. So anyway, if any of you listening haven't completely given up on the idea that you might still have some remaining shred of control over the machine that's sitting in front of you, google "process hacker," and you'll find it. It comes up first of many hits.

Leo: It's on SourceForge, SourceForge.io?

Steve: Exactly.

Leo: Wow. That's awesome. I love it. Yeah, so, yeah, I wonder if, yeah, I wonder if Mark Russinovich is aware of it. That's hysterical.

Steve: Okay. So last Thursday's GitHub blog posting was titled "GitHub Brings Supply Chain Security Features to the Go Community." And it's very short. They said: "The global Go community embraced GitHub from the beginning, both as a place to collaborate on code and a place to publish packages, leading to Go becoming one of the top 15 programming languages on GitHub today. We're excited to announce that GitHub's supply chain security features are now available for Go modules, which will help the Go community discover, report, and prevent security vulnerabilities."

And then commenting on GitHub's announcement, Google's Go Language, also GoLang, product lead, Steve Francia, he said: "Go was created, in part, to address the problem of managing dependencies in large-scale software. GitHub is the most popular host for open-source Go modules. The features announced today will help not just GitHub users, but anyone who depends" - pardon the pun - "on GitHub-hosted modules. We are thrilled that GitHub is investing in improvements that benefit the entire Go ecosystem, and we look forward to more collaborations with them in the future."

Okay. So a little bit of background. The Go Language itself is rapidly gaining ground with 76% of respondents in a developer survey last year, in 2020, saying that Go is now used in some form in the enterprise. So more than three out of four are saying, yeah, we've got Go here.

Go's module system was introduced two years ago, in 2019, to make dependency management easier and version information more explicit. And as a consequence, Go module adoption is also increasing. 96% of those surveyed said that these modules are used for package management, which was a 7% increase over the previous year, 2019. And 87% of respondents reported that only Go modules are used for this purpose. And an overall trend in the survey appears to suggest the use of other package management tools is consequently decreasing. So Go did a good job with package management; and we're seeing, again, inertia is strong. If what you're doing is not broken, the tendency is not to change it. But we're seeing a drift, maybe like in this case 7% a year.

So GitHub's blog posting detailed four primary areas of improvement in supply chain security that are now available for Go modules. The first is GitHub's advisory database, which is an open source repository of vulnerability information containing over currently 150 Go advisories. And that number is growing every day as they curate existing vulnerabilities and triage newly discovered trouble. The database also allows developers to request CVE IDs for newly discovered security issues. GitHub also now provides the dependency graph, which can be used to monitor and analyze project dependencies through go.mod, as well as to alert users when vulnerable dependencies are detected.

And GitHub has introduced Dependabot in this update, which will proactively, and I love this, send developers a notification when new vulnerabilities are discovered in Go modules that affect their projects. I'm sure our listeners all know how great I think it is that we're talking about proactive notification. You know, developers will still need to seriously heed any notifications they receive. But it's not possible to heed notifications that you never receive.

So this is certainly a step in the right direction. And being proactive, I think, that is the future. Automatic pull requests can be enabled to patch vulnerable Go modules, and notification settings have been updated for fine-tuning. GitHub said that when repositories are set to automatically generate pull requests for security updates,

dependencies tend to patch up to 40% faster than when they're not set automatically. And actually, I would think that would be even better than that. But, you know, yay.

So, you know, we've talked, I would guess I would say incessantly on this podcast, about the power that automated and automatic background updates have to rapidly remediate security vulnerabilities. Whenever I do this, I'm reminded by some of our intrepid listeners that with automation comes some risk of subversion. And, yes, that's inarguably true. But, for example, this month alone Microsoft patched 117 flaws, nine of which were zero-days. Four of those were being actively exploited in the wild. As it is, Windows, as we've been talking about so far, is barely holding together. It's not possible any longer to conceive of a world without Windows automatic updates, where as it was once upon a time, every individual end user was obligated to go get and manually install Windows updates onto their own systems. Can you imagine that today?

And those old-timers among us can probably recall the controversy that surrounded Microsoft's decision, back when they made it, to take that job out of our hands. Cranky old guys who were still imagining that they had any real control objected to the idea that their beloved hand-built machines might be changed without their prior approval and oversight. Uh-huh. Well, we had to get over it. That day has long since passed. And I believe we're headed much farther down that path. Like it or not, warts and all, it's the right path for us to take.

In time, I think it's going to become just as clear that the only way for complex, multi-component, multi-sourced software to be built and maintained will be for similar dependency graph-driven automated supply-chain management become standard operating procedure. Developers are just as busy as end-users, if not more so. And yes, it's true that automated supply-chain management brings risk of supply-chain attack. But we're heading in with our eyes open, and this is not the first time we've done this sort of thing. Automating the entire software lifecycle, from the developer's fingers on the keyboard, writing the code, all the way out to code running on machines, even IoT devices, creating the ability to write and publish incremental updates which then securely flow all the way out as binary updates everywhere that code appears, anywhere in the world, is where we need to eventually arrive.

And yes, it will create a mess. It will be a mess of overhead that we do not yet have today. As an analogy, look at what a mess the addition of secure boot has created, an incredible amount of overhead that's beginning to appear in every system to address a problem that almost none of those systems will ever have. But it's there everywhere. And what does it do? All it does, when it isn't being bypassed, is attempt to assure that every step of the operating system boot process uses verifiably signed code. That's it. But it's a mess.

In the future, we're going to muck up our development processes similarly because we have no choice. It's going to be a mess, too. But I think it's as inevitable as was allowing Windows to update itself. It's just not possible, Leo, to imagine a world where the end-users of Windows today are like responsible for keeping their systems secure. Thank god Microsoft did that back then, even though it did annoy those of us who used to know what the files on our hard drives actually did.

One quick note. Someone tweeted whose Twitter name is JF. He tweeted from @jfparis. And he said: "Hi, @SGgrc. I listened to you repeatedly trashing QNAP over several shows. Quality of their software is doubtful, I agree. But unlike many of their peers, it is relatively easy to replace it with a clean Linux distro." And so I just wanted to reiterate that. We've mentioned it before. I wanted just to, since it popped up in my Twitter feed, I thought, yeah, that's worth just reminding people. It's like, you know, if you have a QNAP machine, I would argue, get it off the Internet. I mean, hopefully you've taken that advice a long time ago. But the fact that it can receive a standard Linux or Unix or some

distro means you could give it a well-maintained, far more security-hardened platform for the future, and that it's probably worth doing. And lots more capabilities.

Leo: But in your defense, that would be like not criticizing Windows because, well, you can always install Linux on that PC. You know.

Steve: Well, no. I guess I would not criticize a laptop because you could scrape Windows off of it and put Linux on it.

Leo: Right, yeah. Or desktops, too. Yeah. I mean, but you criticize Windows because it's Windows. And when you talk about QNAP, you're not just talking about the hardware, you're talking about the software, yeah.

Steve: Right, that's a good point. That's a good point. Okay. SeriousSAM & PetitPotam. The first of these two new problems with Windows was just developing, as I mentioned at the top of the show, as last week's podcast was being produced. We noted last week that it had been preliminarily named "HiveNightmare" since we were all in a "nightmare" mode following the many recent printer nightmares of the month. But since then, the name SeriousSAM appears to have taken root, SAM being the abbreviation for Windows' Security Account Manager (SAM). And I prefer this name since the trouble is entirely separate from any printer-related trouble. Unfortunately, "Serious Sam," as you mentioned, Leo, is also the name of a 20-year-old, and still...

Leo: Very good game.

Steve: ...relevant, first-person shooter.

Leo: It's really good.

Steve: Yeah. And googling just the phrase "Serious Sam" will return lots of gaming hits rather than any security information.

Leo: Lot of nostalgia, yeah.

Steve: So anyway, we will take a look at this. Last Monday, on the 19th, security researchers began reporting that the Security Account Manager file on Win10 and 11 was READ-enabled for all local users. That was quite deliberately never true of Windows before 10. It's not good, and it is a critical security mistake.

The Security Account Manager file, as its name suggests, stores sensitive security information including storing and caching all of the hashes for the user and admin passwords the system is aware of. Having this file's security access rights enabled for read access by everyone means that attackers with any access to the system can use this SAM file information, which they should never be able to read, to escalate privileges or access other data. In other words, it's a big no-no.

The next day Tuesday, that is, the following day, the next day on Tuesday, Microsoft immediately issued an out-of-band advisory for this vulnerability, which is now being tracked as CVE-2021-36934. And as of last Thursday, the vulnerability has been confirmed to affect Windows 10 version 1809 and all later, as well as Windows Server 2019 and all later, including 20H2.

A public proof of concept is available that allows non-admin users to retrieve all registry hives. And researcher Kevin Beaumont, who tweets as @GossiTheDog, has released a demo that confirms that it's both possible and practical to obtain local hashes and pass them to a remote machine to achieve remote code execution as SYSTEM on arbitrary targets in addition to privilege escalation. In other words, ow.

CERT's Coordination Center has published detailed vulnerability notes on this CVE titled "Microsoft Windows gives unprivileged user access to system32\config files," meaning that directory. They said: "Multiple versions of Windows grant non-administrative users read access to files in the C:\Windows\system32\config directory. This can allow for local privilege escalation. With multiple versions of Windows, the BUILTIN_Users group" - okay, so that's the group, in terms of the Windows Access Control management, which has been given read-execute permissions to files in that system32\config directory. So it's the built-in users group.

If a VSS shadow copy - VSS shadow copy is the system which allows the snapshotting of an in-use file system so that a moment in time can be captured. Then Windows can continue reading and writing to the file system while that captured moment in time can be compressed or an image can be made of it, or it can be spooled off somewhere and so forth. That's VSS shadow copy. So they said, you know, and also that's the system, like a snapshot is made before you apply a Windows Update every month so that, if it caused a complete collapse of your system, you're able to roll back the changes that were made after that snapshot of the file system was taken.

So they said: "If a VSS shadow copy of the system drive is available" - which almost always will be as a consequence of the fact that they're often being taken - "a non-privileged user may leverage access to these files" - which again they should never have access to - "to achieve a number of impacts, including but not limited to extract and leverage account password hashes; discover the original Windows installation password; obtain DPAPI computer keys, which can be used to decrypt all computer private keys." DPAPI is Windows Data Protection API, which has been part of Windows since Win2000. It offers Windows clients various simple and straightforward cryptographic services. Windows makes use of its own API, this DPAPI, to protect various of its own sensitive local private key stores. But as always, the master key must be around somewhere. This flaw makes it available. And, finally, "obtain a computer machine account which can be used in a so-called 'silver ticket' attack."

Okay. So the VSS shadow copies may not be available in some configurations. However, as long as your system has a drive larger than 128GB, when you perform a Windows Update or use Windows setup to install an MSI package file, a VSS shadow copy will be created automatically to allow a system change roll-back. Okay, so here's some cool things. To see whether your system, any Windows system, has one or more VSS shadow copies available, you can issue the following command from a privileged command prompt. So you'll want to right-click on command prompt, then select Launch or Use As Administrator. That'll give you an administrative command prompt. Then use the command vssadmin (V-S-S-A-D-M-I-N) space list space shadows. It'll say none available, or it'll list them. Okay. So that's the first thing, do you have shadows available?

To check if a system is vulnerable, that is, if your system right now is vulnerable to this, from a non-privileged command prompt, because you want this to fail because you're asking the question without privilege, you type the command icacls, again icacls, and

then the path name to the SAM file. So that would be your Windows directory, typically it's C:\Windows\system32\config\sam. So icacls, space, and then the path to the SAM file.

If it succeeds, it will print out a bunch of stuff ending in "Successfully processed 1 file, failed processing 0 files." Again, from a non-privileged command prompt, that would tell you that your system is right now vulnerable. That is, your built-in users Windows ACL privilege has this read-execute privilege, which you should not have. That was a mistake. If you are told when you do this that that path repeated, and then access is denied, "Successfully processed 0 files, failed processing 1 file," that is, Windows was unable to access the ACLs for that, as it should not be able to, that's good. That means your system is safe. So that would allow you to verify.

There's currently no patch from Microsoft for this trouble, though I'll be surprised if we don't see something soon because this is big. In the meantime, Microsoft did release remediation guidance for Windows 10 and 11 users which mitigates the risk of immediate exploitation of this. For the measures to be effective, it's necessary to first restrict access by changing the ACLS, and then delete any existing shadow copies. Because they will be copies of a pre-ACL fixed system, you don't want anyone to have access to those. If you needed to, you could create a replacement shadow copy immediately after that.

So anyway, the way to do this, and I've got all of this, by the way, in the show notes if anyone can't find it online, you would open up a command prompt or a PowerShell with admin rights. And then you run the following command: icacls, space, and then a path to the config*.*. So C:\Windows\system32\config*.*, then space /inheritance:e. Hit ENTER, problem solved. So it is trivial to fix this problem. So it can't take Microsoft long to just push out something. Maybe they'll even do it again through Windows Defender because they're constantly updating that, and just fix the permissions on all the files in that directory.

Once you've done that, then you delete any volume shadow copies that you may have. It turns out that's also easy to do. I've got that in the show notes. It's vssadmin space delete space shadows, and then some command prompt parameters that I'll let you look up. Again, the very bottom of page 11 of the show notes. Then you probably want to go back and do the vssadmin list shadows to confirm that they're all gone. Now you're safe. And at this point, if you wanted to, you could manually take a snapshot. But on the other hand, next time anything does anything to your system that needs the ability to roll back, you'll have one.

So it would really be interesting to know what happened. You know, was this a change made during development that they forgot to change? Was there something they were going to do where they meant maybe to change one file, but they changed all of them in the directory? I mean, how do you account for this? It's just amazing. But it also sort of suggests that there is a lack of regression testing that ought to be in place. You'd think that a new installation would have regression tests run against it to make sure that exactly this kind of thing hasn't happened. But somehow that didn't happen since the release of Windows 10. And again, this is another one of these problems. Since Windows 10, what was that, that very first build, this has been wrong. And anybody with no privileges could get in, and into something that they should absolutely be excluded from. Yet they've been able to. So is it any surprise that we've been seeing the kinds of problems we have been?

Okay. Next one is PetitPotam, a French-based, I mean, a Paris-based French security researcher whose GitHub handle is topotam, and who appears to have a thing for hippopotamuses, Leo. We'll get to that in a second, yeah. That's from his GitHub page. He recently discovered and went public with, like what day is it, another serious security flaw in Windows which can be exploited to force remote Windows servers to authenticate

with an attacker, thus sharing their NT LANMAN authentication details and certificates. Okay, no, given that this researcher's GitHub page shows a bunch of apparently quite happy and sort of adorable hippopotamuses, it appears that the Potam of PetitPotam is meant to put us in mind of a small hippopotamus.

Okay. So what's the new problem? The trouble surrounds a means of abusing Microsoft's MS-EFSRPC Protocol - EFS as in Encrypted File System, and RPC as in Remote Procedure Call. So MS-EFSRPC. It's the network protocol which enables Windows machines to perform operations on encrypted file system data, stored on remote encrypted NTFS-based systems. But an encrypted file system is not required for this attack to work, just the protocol. The PetitPotam attack proof-of-concept code allows an attacker to send SMB, our good old friend Windows File and Network Printer Sharing and all that over 445 port, requests to a remote system's MS-EFSRPC endpoint interface, to cause the target victim computer to initiate an authentication procedure and thus share its authentication details.

Attackers can collect this data and abuse it as part of an NTLM relay attack, you know, NT LANMAN, right, LAN Manager, relay attack, to gain access to remote systems on the same internal network. PetitPotam cannot be exploited remotely across the Internet, thank god, you know, thank goodness. It's an attack that's designed to be used inside large corporate networks where attackers could use it to force domain controllers to cough up their NTLM password hashes or authentication certificates. This could then in turn lead to the complete takeover of a company's internal network. So we can see why Microsoft responded with surprising speed to the new threat created by this public publishing of a proof of concept.

Okay. So there's a related exploit using MS-RPRN, okay, MS-RPRN. That's their print service remote protocol, though the developer of the PetitPotam exploit tweeted Sunday before last on the 18th when it first pointed the world to his discovery. So he said: "Hi all. MS-RPRN to coerce machine authentication is great, but the service is often disabled nowadays by admins on most orgs. Here" - and this is the birth of the PetitPotam. "Here," he tweeted, "is another way we use to elicit machine account auth via MS-EFSRPC. Enjoy!! :)." And then a link to his GitHub page. This was earthshaking.

In response to that, four days later, another researcher replied: "Finally finished testing it. It's quite brutal! Network access to full Active Directory takeover." He said: "I really underestimated the impact of NTLM relay on PKI #ESC8. The combo with PetitPotam is awesome! Everything is already published to quickly exploit it." Which, yeah, is exciting these guys, but it's a disaster for the enterprise. "Tests carried out by multiple security researchers have shown that disabling support for MS-EFSRPC did not stop the attack from working. It has been tested against Windows Server 2016 and Windows Server 2019 systems, but security researchers believe PetitPotam impacts most Windows server versions supported today." In other words, so far it has affected all that they've tested.

Florian Roth, the Head of Research at Nextron Systems, was quoted in The Record saying: "The problem with this type of attack is that it will take a considerable amount of time and consideration to develop appropriate countermeasures. These are design flaws that are more difficult to fix. It's much easier to just patch a vulnerable font driver DLL or Internet Explorer library." In other words, we're here again with a fundamental flaw in a core Windows protocol where whatever fix is found must also not break existing facilities. Again, it's not a bug, it's a feature. But it's a bad feature.

The TrueSec, TrueSec.com, has a blog posting where they describe in a little more detail exactly how this happens. And it will certainly make any of our corporate listeners sit up and take notice. Hopefully all that they'll be doing when they're hearing this on the podcast is going, yeah, yeah, yeah, did it all right. Here's what TrueSec said: "This advisory" - that is, theirs - "is related to the recent Certified Pre-Owned whitepaper

discussing the possible abuse of the Active Directory Certificate Services," and then they said, "AD CS role in combination with Credential Relay Attacks such as MS-RPRN and the more recent MS-EFSRPC aka PetitPotam.

"The MS-EFSRPC protocol can be used to coerce any host, including Domain Controllers, to authenticate to a specific destination. The designated destination then forwards the NTLM" - again, NT LANMAN - "credentials to another device that is configured to accept the Domain Controller's certification, resulting in an abuse of those services. An attacker can target a Domain Controller to send its credentials by using the MS-EFSRPC protocol and then relaying the DC NTLM credentials to the Active Directory Certificate Services AD CS Web Enrollment pages to enroll a domain controller certificate. This will effectively give the attacker an authentication certificate that can then be used to access domain services as a domain controller and compromise the entire domain.

"AD CS is especially interesting as it offers role services that by default accept NTLM-based authentication." Let me repeat that because that's going to come back in a minute. "Active Directory Certificate Services is especially interesting as it offers role services that by default accept NT LANMAN-based authentication." Then they finish: "The Certificate Authority Web Enrollment and Certificate Enrollment Web Service can be abused to issue certificates by performing NT LANMAN Relay Attacks using MS-EFSRPC, MS-RPRN or other API that offer similar behavior."

Okay. So I'm going to conclude with a couple of points and observations. First, this is of no concern for end-user Windows people.

Leo: Oh, should have said that upfront. Oh, geez. Could have saved me a lot of acronyms.

Steve: This is entirely a high-end enterprise worry when an organization is running with Domain Controllers and Active Directory Certificate Services. So if you don't already understand all of this, if you don't understand that you're not vulnerable, I mean, I'm sorry, if you don't...

Leo: You're definitely not.

Steve: ...already understand that you are - if you don't understand that you might be vulnerable...

Leo: Right.

Steve: ...you definitely aren't.

Leo: Definitely not, yeah.

Steve: Right.

Leo: If you don't know what a domain controller is, you're fine.

Steve: You're fine. Second, when exploited, it provides a means for an attacker who is already present on the victim's network to fully compromise and take over the entire operation. That's obviously not good because it completely collapses all security containment, enterprise-wide. But it is at least constrained to be a local-only attack.

Okay. Now, in response to this, Microsoft immediately blamed their old, but still widely used and enabled-by-default NTLM (NT LANMAN) protocol. I have a link to what they said in the show notes. I've excerpted for the podcast. "Microsoft is aware of PetitPotam, which can potentially be used in an attack on Windows domain controllers or other Windows servers. PetitPotam" - this is Microsoft writing - "is a classic NTLM Relay Attack, and such attacks have been previously documented by Microsoft, along with numerous mitigation options to protect customers."

Continuing, they write: "To prevent NTLM Relay Attacks on networks with NTLM enabled, domain administrators must ensure that services that permit NTLM authentication make use of protections such as Extended Protection for Authentication (EPA) or signing features such as SMB signing. PetitPotam takes advantage of servers where the Active Directory Certificate Services (AD CS) is not configured with protections for NTLM Relay Attacks. The mitigations below outline to customers how to protect their AD CS servers from such attacks." And then Microsoft goes on to describe what needs to be changed, disabled, and blocked to thwart this attack.

Okay. Now, note that Microsoft calls it a "classic NTLM relay attack," as if everyone should be aware that NTLM, which they once believed to be fabulously secure, has become so completely broken that attacks on it are considered to be classic, like Coke.

Leo: Oh, yeah, of course.

Steve: And that as a result of NTLM's fully acknowledged crap security, it should never be used unless there's really no other choice. Except for this. Now, I'm quoting an imaginary, fully forthcoming Microsoft saying: "You just bought a brand new server for your enterprise, and we've just installed a bunch of old and insecure crap on all of your servers, just in case they might need to connect to something else that you may have lying around that's also old and insecure. To eliminate any confusion about why all this complex stuff might not be working, we turned everything on, and it's fully enabled, so that your shiny new systems would just work out of the box without you needing to learn anything about them, or even wonder for a minute why you couldn't just plug everything in and have it all go. So it does. Because we're Microsoft, 'Vulnerable by Design.'"

Leo: Oh, lord. Sigh.

Steve: Leo, what's sad is it's true.

Leo: Yeah, there's no reason why that should be turned on.

Steve: It's true.

Leo: Probably not even included.

Steve: I'm not making this up.

Leo: If you need it, you could download it.

Steve: No, no, it's included.

Leo: Yeah, that's ridiculous.

Steve: That's why everybody has it. That's why it's a big worry. Microsoft literally has their systems designed with everything on because they don't want a phone call. They don't want somebody wondering what this error message means. They just want it to all work. So it's up to the admin to go in and, oh, yeah, what about those classic NT LANMAN relay attacks? Shall we do something about that? Are they still classic?

Leo: Oh, god.

Steve: It's just painful.

Leo: Yeah. No, you're absolutely right. There's no reason to include it, let alone include it and turn it on, when 90% of people never use it.

Steve: Enabled by default, Microsoft said in their bulletin.

Leo: Well, it keeps people in business. That's why we can never stop hiring IT professionals. So that's a good thing.

Steve: Like I said, the digits this podcast requires are going to push that little tag at the bottom of the screen right off to the right, Leo. I'm going to come up with - because the episode numbers are going to have so many digits there won't be room for...

Leo: Just a number. That's all we need. Just a number.

Steve: That's right.

Leo: Steve, you've done it again. A number of people in the chat are saying, "And this is why we listen to Security Now!." You're absolutely right. It's unconscionable. It's unconscionable. Steve does his thing at GRC.com. That's his website. That's where you'll find SpinRite, world's best mass storage maintenance and recovery utility.

Steve: It's rolling right off your tongue, Leo.

Leo: Yeah. It's currently at 6.0. 6.1 is coming. You can participate in the development and get a free copy if you buy right now, GRC.com. While you're there, check out all the other things Steve has for you, including this show, 16Kb as well as 64Kb audio versions of the show; really nicely written transcripts, so you can read along while you listen, or search, which is a great feature. He also has a feedback form there, GRC.com/feedback. But another way to get him, and I think probably the easiest would be to slide into his DMs on Twitter. His Twitter handle is @SGgrc, and his DMs are open.

We have 64Kb audio plus video versions of the show at our site, TWiT.tv/sn. You can also subscribe in your favorite podcast player. That way you'll get it automatically. In fact, do me a favor. If you're doing that, leave a five-star rating and a review for us. Let the world know about Security Now!. It's very helpful to us. We also have a YouTube channel dedicated to Security Now!. Lots of ways to consume it. Make it easy.

We do this show Tuesdays, right after MacBreak Weekly. That's usually around 1:30 Pacific, 4:30 Eastern, 20:30 UTC. You can watch us do it live. There's a stream, audio or video, at TWiT.tv/live. If you're watching live, of course, it's great to chat live. There's a free chatroom at irc.twit.tv, along with a Discord chatroom. And we'd love to see you join us during the live program. The chatroom is always a great part of the show. Steve, have a wonderful week, and I'll see you next week.

Steve: Thank you, my friend. Right-o. Bye.