Blacksmith

Description: This week we look at a critical 9.8-rated vulnerability affecting Palo Alto Network's widely deployed VPN Firewall appliance, and at a welcome new micropatch from the 0patch guys, the nature of which leads me into a bit of philosophical musing about the Zen of coding. We're then rocketed back to reality by a review of last week's Patch Tuesday, looking at what it broke and happily what more it fixed, including hints that Christmas might finally be coming to printing by December. We have some more encouraging ransomware vs. the law news, and we examine the question of how to make big money defrauding online advertisers. I'll then share some fun and interesting closing-the-loop feedback from our listeners and an update on my SpinRite work. Then we're going to take a look at "Blacksmith," the evolution of Rowhammer attacks on DRAM.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-845.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-845-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to talk about Patch Tuesday, 55 flaws. Steve will break it down. There's a security issue in a piece of security hardware you're going to want to know about. And then the latest version of Rowhammer. It's called Blacksmith, and it affects all DRAM everywhere. It's all coming, oh boy, it's coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 845, recorded Tuesday, November 16th, 2021: Blacksmith.

It's time for Security Now! the show where we protect you and your loved ones in the vast online world of cyberspace with this guy right here, who's been around since before cyberspace began. What are you doing, Steve Gibson?

Steve Gibson: It all is blowing up, Leo. Just...

Leo: It's blowing up.

Steve: ...kaboom, yes.

Leo: Security Now! time. I know you wait for this all week long. So do I. Here we are. Hello, Steve.

Steve: Well, hello, Leo. Hello.

Leo: Hello.

Steve: So we have actually, the topic I was going to discuss got bumped to next week. I was going to talk about something called HTTP Request Smuggling, which it turns out has been a known problem for a while; but it's only just become, like, abused. So I thought, oh, that'll be fun to talk about. But then yesterday a paper dropped from some guys at COMSEC who are the guys at ETH, I want to say Yurich, but that's not right.

Leo: Zurich.

Steve: Zurich. Hello. Yes. Zurich. The security guys who've done, you know, we've referred to them a bunch through the years. They've taken Rowhammer, the original DRAM Rowhammer problem, way further. And it's not good. So I thought, okay, that's more important than a, well, a problem which has just come into fashion, apparently. So we have a lot of things to talk about. We're going to look at a critical 9.8-rated vulnerability affecting Palo Alto Networks' widely developed VPN Firewall appliance, and at a welcome new micropatch from those 0patch guys. And the nature of it leads me into a bit of a philosophical musing about the Zen of coding, which had me thinking of you because I know that - and also a little bit about chess and your affection for that. But anyway, we'll get there.

Leo: Boy, my two favorite subjects. I can't wait.

Steve: Then we're rocketed back to reality by a review of last week's Patch Tuesday, looking at what it broke and, happily, what more it fixed, including hints that Christmas might finally be coming to printing by December. But not yet, children. We have some more encouraging ransomware vs. the law news, where the law apparently is winning. And we examine the question of how to make big money defrauding online advertisers. Then I'll share some fun and interesting closing-the-loop feedback from our listeners. And, boy, have we got some neat listeners. A brief update on my SpinRite work, and then we're going to take a look at Blacksmith, which is the evolution, probably the ultimate evolution of Rowhammer attacks on DRAM and the fact that nothing can be done.

Leo: Oh, no.

Steve: It's one of those happy endings.

Leo: Oh, no. And I hope you'll do a refresher on what Rowhammer is and all of that.

Steve: I've got that in the notes, yup, for sure.

Leo: Because there are so many things; and, you know, we forget.

Steve: And then we do have, for those of us who have, well, not us, I'm no longer among them, but maybe Elaine, who she uses a satellite link, last time I checked, and her connection is slow, which is why she originally asked me if I could recompress these from your beautiful-sounding 64Kb down to 16 to conserve her bandwidth. Well, there is a solution, it turns out, Leo, for people with highly constrained bandwidths who wish to install Windows 11. And we've got the Picture of the Week to demonstrate that.

Leo: Okay. No hints. No clues. But that's good. That's coming up. Ready for the Picture of the Week. You have this picture, practically.

Steve: Yeah. So someone tweeted this to me, and it just caught me at the right moment. It was kind of funny because those old-timers among us recall when you did install Windows, like 3.1 came on a deck of 3.5" floppy disks.

Leo: Show them your 8" Windows 1.

Steve: So, yeah, I actually just...

Leo: Look at that.

Steve: It just occurred to me to pull those out of a drawer. They weren't even giving it a version number at that point because it was just called Windows.

Leo: Wow.

Steve: And of course it used to run on DOS. So it wasn't its own operating system. So the label says Windows Standalone Runtime System Setup Disk 1 of 2 and Utilities Disk 2 of 2, copyright 1984 and 1985.

Leo: Nice. Just to bring us up to the present, though.

Steve: Yes. I was going to say, since then, Leo, there's been a little, I don't want to say "bloat," well, maybe I do. We now have Windows 11 available on 3.5" disks. This shows two of them of 3,852.

Leo: Did you do the math to see if that's accurate? I bet that's low.

Steve: Actually, in my head it's about right because you figure one of those is 1.44MB. So 1.44MB times 3,852, that's going to bring you to around the, what, 5GB or so.

Leo: Yeah, that's about right, yeah.

Steve: Which is where Windows 11 is now. Now, of course it was always a problem, back when we were installing Windows from 3.5" floppies, is that you'd get about to the fourth disk, and it would go, eh eh eh eh and, like, you'd have a read problem.

Leo: Oh, god, I remember those days.

Steve: So you'd eject it and put it back in again.

Leo: Ugh.

Steve: And hope that you'd get like a better settling of the hub, which was metal on these. The track density of these was high enough that you couldn't use just a standard Mylar hole. So they had metal hubs in order to get better centering. But even so, eh eh eh eh eh. And it's like, oh, crap. Now, given that installing Windows 11 this way would require 3,852 of these, I would say your chances of actually getting Windows 11 running are about the same as they are today when you download it and run it. It's like, good luck. Anyway, just a fun Picture of the Week, and thank you to whomever thought to send it. I appreciate that.

Leo: That is so true, too. Gosh. Thousands.

Steve: Yeah. Okay. So around 10,000 VPN Firewall appliances from Palo Alto Networks are vulnerable. And once again, actually, I don't blame the tech press for calling it a zero-day because the guys who found it called it a zero-day in their announcement. So the tech press, who is, you know, just taking their copy from the announcement, parroted that; right? They're calling it a zero-day. Nothing about it is a zero-day, but it's definitely a very serious vulnerability with a vulnerability severity score of 9.8. And as we know, we're still waiting for a 9.9 or a 10. I don't know what that even looks like because everyone seems to hit the ceiling at 9.8. But this is as bad as it gets, 9.8.

It affects, as I said, Palo Alto Networks. The product is the GlobalProtect firewall, which allows unauthenticated remote control and remote code execution on multiple versions of their PAN-OS from a family version 8.1, prior to 8.1.17. So that's where they fixed it, in 17. If you've got 8.1 less than 17, you're in trouble. And this affects both their physical and virtual firewalls. The discovery was made by a group we've never referred to before known as Randori, R-A-N-D-O-R-I, and their site is Randori.com.

Their disclosure, which as I said also sadly labeled this as a zero-day, they wrote: "The Randori Attack Team developed a reliable working exploit and leveraged the capability as part of Randori's continuous and automated red team platform. Our team was able to gain a shell on the affected target, access sensitive configuration data, extract credentials, and more. Once an attacker has control over the firewall, they'll have visibility into the internal network and can proceed to move laterally."

And of course this is a problem because there are 10,000 of these things out there, all vulnerable until they're patched, and we know how that goes. And, you know, no mom-and-pop op is going to - I don't have one of these Palo Alto Networks firewalls. I've got a nice little pfSense system here. The point is that these are going to be serious enterprises that are using these Palo Alto GlobalProtect VPN firewalls, so they're going to be a hot target for guys who want to get in and get up to some mischief, moving laterally to do their nefarious ransomware stuff.

"So in an effort to avoid enabling misuse, technical details related to this" - it's been given a CVE number 2021-3064 - "will be withheld," they write, "from public dissemination for a period of 30 days from the date of this publication." And again, that's why it's not a zero-day. I mean, it's not in the wild. It wasn't discovered being used. It's a vulnerability. So, okay, good that we found out about it. They responsibly disclosed this at Palo Alto Networks, who quickly said, after recovering from their unhappiness, updated this 8.1.16 to 8.1.17, fixing the problem. I'm sure they sent out announcements to anyone they had access to, hopefully stressing the importance of updating the appliance.

Leo: So it's kind of a security success story.

Steve: Yeah.

Leo: I mean, nothing's perfect. You're going to have flaws. You wish you wouldn't.

Steve: Yes, yes. And as always, it's that last mile is the problem; right? It's a security success story until we start covering the disasters which ensue when people don't patch; when they don't, like for whatever reason, communication failure, lack of resources, the IT security guys are too busy putting out other fires, you know, maybe the enterprise hired a consultant. I hope they have somebody full-time on staff because that's now, like, required these days. But what will happen is more than half of these will not get updated. And Rodan has found them all. These have been indexed so that bad guys can immediately target them, use unpatched versions to exploit them and then crawl in. So, you know, yes. It's a success story all the way, as I said, Leo, to like that last mile, which requires that people get these things kept current.

Leo: Shodan. Shodan, not Rodan. Shodan.

Steve: Oh, Shodan.

Leo: No, I understand where Rodan came from because that's kind of sort of like the people who discovered this. So that's what confused me at first.

Steve: Right, Randori, right.

Leo: Yeah, Randori, yeah.

Steve: So anyway, this question of should things update themselves autonomously. You know, we're now, you know, the machines that we have in our pocket, they do that. Windows has continued to push that further. They have their monthly patch cycles; but now they've, like, started doing this other new thing because that's not in some cases often enough. So they're able to update their Windows Defender patterns at any time they want to, and so block things that are coming in before Windows has had a chance to be strengthened against them. So, you know, the fact is we're seeing this model evolve where updates are being delivered autonomously.

And it seems that there's still pushback. When I talk about this for consumer routers, I get, not hate mail, but people saying, you know, think of what happens if a mistake is made. And it's like, well, yeah. But is that worse than a serious vulnerability being left unpatched by the bulk of the population? Consider what the chances are of a consumer router ever being updated. I mean, our listeners, yeah. Pretty much more than that? Probably not.

Anyway, the vulnerability chain consists of a method for bypassing validations made by an external web server, and that's where this term HTTP Smuggling comes in. This is actually an HTTP Request Smuggling attack which we will be covering in detail next week. We've never talked about it, and it's interesting. And obviously here's an example of a really bad problem being impacted that way. The good news is the Randori guys are acting responsibly. They're keeping the details to themselves. I'm sure they'll come out with full details in a month. In the meantime, boy, if you or anyone you know is using a Palo Alto Networks VPN Gateway, make sure that you're patched.

The 0patch guys, who I think have found themselves a niche in the world, have produced a micropatch. Recall that we recently discussed a worrisome local privilege escalation vulnerability in the Windows user profile service which affected, or I should say unfortunately affects, all versions of Windows. This was that one that Abdelhamid Naceri discovered and privately, responsibly, properly disclosed to Microsoft months ago. They then listed this as patched and fixed in August. But after he reverse engineered the change that Microsoft made to see what it was, he was kind of curious, oh, cool, how they fixed this.

He discovered that the only thing fixed was that the simple proof-of-concept demo he'd provided to demonstrate the more severe underlying problem no longer worked. The underlying problem remained. So it wasn't fixed in August when it was supposed to be, nor in September or October, or last Tuesday, in November. On the one hand, if it's exploited, it allows an attacker to elevate themselves to full system, you know, root privilege under Windows. But on the other hand, any such attacker would need to have knowledge of some second credential that would work on the same machine.

The CERT/CC guy whose name escapes me, I've talked about him before, he commented that that would - and he correctly said - Will Dormann, that's his name. He correctly said that that meant this wouldn't be as widespread as many other things. Naceri reminds us that any domain credential would be sufficient. So that does somewhat lower the bar for an attacker. I won't comment on the fact that when BleepingComputer asked Microsoft about this still-unresolved problem, BleepingComputer was told that: "We're aware of the issue and will take appropriate action to keep customers protected."

Of course, okay, I said I wouldn't comment. But what I will comment on is that last Thursday, after seeing that Microsoft still hadn't fixed this for the fourth month in a row, the guys over at 0patch created one of their very cool little micropatches that, believe it or not, actually works and solves the problem. It can be applied to any Windows system needing interim protection until Windows gets it fixed, and of course potentially on older systems that Windows will no longer fix. That's really sort of the niche that the 0patch guys have found is you have to subscribe. You can sign up for free and get this for free, and it will be available until Microsoft fixes it for free.

But you can also, you know, if you're wanting to keep older systems alive for some reason that Microsoft isn't fixing, these guys are still fixing all these things. So you can subscribe, and this micropatch system automatically updates itself. You know, and the patches are, wow, Leo, 23 bytes. So, yeah. Anyway, they're available for 32- and 64-bit versions on any Windows 10 system with at least the October or the November updates in place. So again, let's hope that Microsoft gets this fixed in December. Be a nice

Christmas present for Windows users. Until they do, this little micropatch is free. You do have to register, however.

And I haven't taken any time to research in detail what it was that Naceri found and what's going on. But the founder of 0patch did say that while this issue could in theory also impact earlier Windows versions, he said: "The vulnerable code is different there, making the window for winning the race condition extremely narrow and probably unexploitable." Well, that's interesting, a race condition. We've on this podcast never had the occasion to discuss race conditions. But it's a common problem that can crop up any time multiple things might be happening at once, where the precise sequence of them happening is important.

A few weeks ago I mentioned the counter/timer in the original PC which used what was known as a "ripple counter." A ripple counter in digital logic is a series of individual toggling flip-flops connected in a chain. When a single toggling flip-flop's input goes to logic "1," nothing happens except it registers it internally. But when that input drops back to logic "0," the flip-flop flips to its other on or off state, whichever the other one is. And if the output of that flip-flop is connected to the input of a next flip-flop, you start to obtain a binary counter, where the number of bits in the counter is the number of flip-flops in the chain. And I should confess, this is how I spent my youth was like drawing these things and drawing logic diagrams. And if anyone wants an interesting challenge, implement a toggling flip-flop with mechanical relays. Even if you don't actually build it, just draw them because I did, and it's an interesting challenge.

Anyway, the problem is flip-flops in the real world do not toggle instantly. They have what's known as a propagation delay, which means that their output changes just a little bit after their input goes to zero. The flip-flop's internal logic takes a bit of time to sort things out. This means that when the input at the front of this ripple counter goes from "1" to "0," which will start to update to the next count, the changes in the count might "ripple down" through the length of the counter.

Okay. Now let's say that you want to take a snapshot of the counter's current count, but you have no idea when that counter might be counting. This is where we get into a perfect example of a race condition. If we happen to take a snapshot of this binary ripple counter while that new count is rippling down the length of the counter due to the propagation delay in each of the counter's stages from one stage to the next, we'll obtain an erroneous reading due to a race condition which we've created between the counting event and our sampling of that counter.

And race conditions aren't just electronic. They can occur socially, too. They're a favorite vehicle of screenwriters who have one person racing to answer the phone before someone else gets the chance to do so, to avoid some embarrassing outcome, a race condition in real life. In electronics and in software, these race conditions can be quite tricky to spot, and they've been known to introduce some of the hardest-to-find bugs there are. The reason may be that they may not occur predictably or even often, but just maddeningly. For example, the state of that ripple counter might be properly sampled thousands or hundreds of thousands of times before the sampling just happens to catch the counter in the process of rippling.

The other maddening thing is that the act of observing the system might change it. For example, adding the tiny capacitance of a high-impedance oscilloscope probe when checking a high-frequency signal can change its timing just enough to prevent the problem from occurring while you're looking at it. And the same goes for debugging code. If you step through the troublesome area of some time-sensitive code, everything works while you're watching it. In fact, the problem only occurs, for example, on odd Tuesdays when the moon is full and the mailman is late. Otherwise everything works fine.

So our lesson here is the need - I guess first the appreciation of or appreciation for and then the need for true vigilance. In any asynchronous multi-threaded application where more than one thread might share access to variables or objects, always consider every implication of that very carefully. Always keep the possibility of these sorts of collisions front of mind, and work very hard to never create such problems in the first place because it's far better to prevent them than to have to go back and try to find those gremlins later.

And as I was thinking about this, about the nature of the need to appreciate how big a problem this is, I sort of got into thinking about new coders who, you know, anyone starting out at the beginning is going to begin their craft without any appreciation, because you can't have any in the beginning, for the subtleties and nuances of the way complex coding problems are best solved. And the word "best" is critical here. For newbies, it might at first seem that any solution to a problem with computers is as good as any other. After all, they're all solutions; aren't they? But with time and patience and lots of practice at coding, a coder gradually matures to learn that there are superior, more elegant, and more robust ways to think about, to frame, and then to solve a problem that may have many solutions. Most coding problems do have many solutions. Yet it's often the case that there really is one best way. And I think that the great joy of coding comes from the satisfaction of knowing that you have found the best solution.

I'll often code something, and then I'll wonder, having done so, like I'm done, if I can't find a better solution. What I think happens is that the act of solving the problem the first time teaches us something about the problem that hadn't been initially appreciated. For example, I might have needed to add an awkward bit of code at the end to handle an edge case that I hadn't seen coming. But if I'd appreciated it at the start, as I do now at the end, might I have come up with an overall superior solution that inherently incorporated and didn't need that special case handling because it solved the problem the right way?

So the point is I may have just learned something about the problem that might now suggest an entirely different and superior approach. So what I often do is I'll leave what I wrote in place so I can go back to it if my intuition is wrong, and I don't find a better way. I'll open up a big region, a blank space in my editor, and immediately tackle the same problem again, but this time starting with a deeper understanding of the problem I'm trying to solve. And for me that's the continual fascination that I have with coding. It's why I enjoy it so much is 50 years later, 50 years after having started this, I'm still learning things because I'm tackling new problems. I'm taking all of this experience that I have, but I'm finding that I can still recast a first solution in something better the second time.

And it's true. It will make no difference in what that chunk of code does. But it might make a difference in the way it does it. One solution may be aesthetically cleaner and superior to the other. And as utterly abstract as code can be, within its own little microcosm it really is possible to create little works of art, little constructed gems. And I know that not everyone obtains fulfillment from building things, and not everyone is as curious as I am. And not everyone may have the luxury of taking the time to find that one best solution. If you are those things, then what I've just articulated makes all the sense to you in the world. And if you aren't those things, maybe now you can understand a little bit better those of us who are so weird in this particular way. And Leo, I have a feeling that you may be one of them.

Leo: I'm in that former group, you bet. It's one of the things I go home and can't wait to get going on, yeah.

Steve: Yeah. There just is something magical about it.

Leo: You know, race conditions are a big problem these days because we do concurrent programming so much.

Steve: Exactly.

Leo: We have multiple processors. So it's not at all unusual to have multiple processes running at the same time. I have to say, though, I think some of it is because we made a big mistake many years ago going with imperative programming and programming with side effects. If you don't have side effects, you don't have problems with concurrency. You don't have dependencies. And, you know, I guess sometimes you need side effects, for sure you do, with IO and so forth. But it seems to me that some of this comes from just a mistake made early on in how we write code. It's why I'm a fan of functional programming.

Steve: I think that's probably true. As you were suggesting that it was fundamental, I was thinking of, for example, I have some code which is manipulating a list, and where I'm inserting an object into a list of objects. So I'm in the process of taking the pointer that points to the next one out and replacing it with a pointer that points to the new one.

Leo: Exactly, yeah.

Steve: And then putting the pointer that was pointing to the last one at the end of the new one, inserting that into the list. Well, that's a fragile moment in time.

Leo: Exactly. Right. And if another process is counting on that list, it's in an unknown state.

Steve: Yes, yes. So if something else, another thread in the same process or, if it's a cross-process object which is available, it might go and take - it might get swapped in, time sliced in, and be attempting to traverse that list while I'm in the process in a different thread of maintaining it.

Leo: Exactly, yeah.

Steve: And that's another good example of something that might work for three years and then just blow up. And it's like, what happened? And again, it's like just a moment in time where at that race condition between two threads trying to simultaneous access. And so there are solutions in Windows. It's called a "critical section."

Leo: Right.

Steve: You're able to request a critical section of code that no two threads are allowed to be in at the same time. And so in my own list management code, I grab a critical section, do the work, and then I exit the critical section.

Leo: You lock it, in effect, yeah.

Steve: Basically lock that little block of code.

Leo: That's why it's called a race, because the winner of the race can vary, and whoever wins will get a different result. Ideally, with code, you want the same inputs to produce the same outputs every single time.

Steve: Determinism.

Leo: It's determinist. And to the degree that it's not is the degree - there are languages like Erlang that are designed to handle concurrency specifically for this. I think if Windows critical code worked better, we would see less of these bugs. But apparently either people aren't doing it or, you know, maybe it's sloppy coding. I don't think it's always sloppy coding. I think sometimes it's just you don't really realize the side effects that you're interested in.

Steve: Well, okay. To take Microsoft's side for a minute, can you imagine touching the code of Windows?

Leo: Right. It should all be critical section, the whole thing. Do not touch.

Steve: I mean, I have the advantage of being a one-person team who wrote everything that I'm doing and touching. So if I make a mistake, it's mine. At the same time, I'm less likely to because it's mine. Nobody wrote anything other than I did. And so I'm not looking at code thinking, okay, I think - long time ago you and I talked about some of the fundamental problems of coding. One of them is bad variable names; right? You name something badly, and then later you think that it's a good name. But if it's a bad name, it doesn't really describe what you think it does. And so that's one of the problems.

And the other is like fuzzy definitions. If you're not really sure of what some code does, you're in trouble because the computer's not sure either. And if you're not sure, it's not going to be any more sure. And so there are, like, again, as you spend time coding, you end up sort of with like your group of things that you learn to watch for. And I've learned the importance of being very careful about what I call things because later I'll come back, and I'll think they're exactly what I called them. And if I'm wrong, I get myself in trouble.

Leo: Yup. Well, and you also get some sympathy from people who make mistakes because it's a hard and a high-stakes practice. It's not easy.

Steve: Which is what makes it fun.

Leo: That's what makes it fun, yeah.

Steve: So we did have last Tuesday Patch Tuesday. We already know what Patch Tuesday didn't patch; right? It didn't patch Naceri's problem. So what did it patch? It received fixes for a total of 55 flaws in Windows, two of which were zero-day vulnerabilities in Exchange and Excel, both under active exploitation in the wild. And by coincidence, the zero-day in Exchange had previously surfaced during that recent Tianfu hacking contest. Four other vulnerabilities, although not being used in the wild, also qualify as zero-days because, while they are not under active exploitation, they were first learned of through public disclosure without any available patches ready or even started. So of the total of 55, six are classified as critical, the other 49 as important.

The exploitation breakdown is that 20 were elevation of privilege vulnerabilities. And we know now, even though they don't seem as scary as remote code, that bad guys want them because they can often get in as an unprivileged user, then elevating themselves to system is what they need to do in order to escape from the containment that the operating system provides. So 20 were elevation of privilege; 15 were remote code execution vulnerabilities, gulp; 10 information disclosure; four spoofing; three denial of service, basically means that you can crash something, but that's about it; and then two were security feature bypasses.

The Excel flaw, which was the other of the zero-days that wasn't the Tianfu discovery, was discovered by the Microsoft Threat Intelligence Center and has been actively used in malicious attacks. The four publicly disclosed vulnerabilities, those other four that technically are zero-days because they were posted publicly, have not been seen being exploited. Two were Windows RDP (Remote Desktop Protocol) information disclosures, and the other two were 3D Viewer remote code execution vulnerabilities. So that's the good news.

November did break something; but, boy, you'd have to be deep into Azure authentication innards to understand what. After installing last Tuesday's updates, Microsoft said that users might experience authentication problems on Domain Controllers running Windows Server. Okay, that we understand. They explained, and I'm just going to quote them because I only have a vague idea of what this is. Well, maybe better than that, but still.

"Kerberos authentication," they wrote, "will fail on Kerberos delegation scenarios that rely on the front-end service to retrieve a Kerberos ticket on behalf of a user to access a backend service. Important Kerberos delegation scenarios where a Kerberos client provides the front-end service with an evidence ticket are not impacted." Oh, I guess that's good. "Pure Azure Active Directory environments are also not impacted by the issue."

And then, as if to clarify, on the Windows Health Dashboard Microsoft wrote: "After installing the November security updates, you might have authentication failures on servers relating to Kerberos Tickets acquired via S4u2self. The authentication failures are a result of Kerberos Tickets acquired via S4u2self and used as evidence tickets for protocol transition to delegate to backend services which fail signature validation." So there. The good news is, whatever it was that broke, they fixed it yesterday, Monday, yesterday, with the release of an out-of-cycle update. So if anyone had updated last Tuesday, and their use of Kerberos authentication to Domain Controllers had died, it's now fixed as of Tuesday. You've got to go get it, probably. So you can do so.

On the Windows 11 front, Windows 11 received KB5007215, a cumulative update to fix security vulnerabilities and bugs newly introduced, or at least occurring, in previous versions. This update is mandatory because it contains security updates, performance

improvements, and bug fixes for Windows 11 21H2. After installing the update, Windows 11 will have its build number changed to 22000.318. So that's one you want to look for.

There was, it turns out, a newly introduced problem with GDI+, which prevented the proper display of UI elements under Windows 11. That's fixed, and GDI+ applications should now be working. And the update also fixes that mysterious, weird, built-in application signature certificate chain verification bug which was preventing a subset of apps - remember, not all, but a few, and some could be fixed, but some others couldn't, such as the Snipping Tool - from working in some instances for some users located on some planets. Now everything is supposed to be working correctly once again for everyone after KB5007215.

Oh. And finally, Microsoft also stated that they made changes to the servicing stack to resolve bugs or issues preventing Windows Updates from properly being installed. So now, presumably, Windows Updates will no longer be prevented from being properly installed, to the tremendous relief of all concerned.

And December promises to be Christmas for Printing and more. It appears that the raft of continuous, embarrassing, and unrelenting problems that Windows has been experiencing has caught Microsoft's attention. Yay. Late last week they released a new build for Windows 11 Insiders in the Beta and Release Preview Channels, and it promises to have fixed a long list of issues impacting Windows 11. And Leo, I imagine you and Paul and Mary Jo will be talking about that tomorrow.

Among the significant bugs addressed in this build, which is 22000.346, are fixes for known issues preventing USB print devices with support for Internet Printing Protocol Over USB from completing the installation and leading to some USB print installers reporting that they don't have a printer after the printer's plugged in. So there's that.

The other issue fixed is one that we talked about a while ago, several weeks ago, and that's where the three weird bugs - RPC_S_CANNOT_SUPPORT, ERROR_INVALID_LEVEL, and ERROR_INVALID_PRINTER_NAME - those were occurring to some users who were trying to print remotely over network shared printers. Those are going to be fixed, well, have been fixed in this Beta and Release Preview. And it's expected to roll out to all Windows 10 and 11 customers during the December 2021 Patch Tuesday. So yes, everybody's going to get their printer things fixed, finally.

So the U.S. has detained a crypto-exchange executive for helping the Ryuk ransomware gang launder their profits. I'm not talking now any longer about the infinite series of ransomware attacks. Maybe if something of note that goes onto the popular news, we'll talk about it, if it's like huge like Kaseya or one of the huge historical attacks. But I did want to keep everyone in the loop about news of the continuing U.S.-led pursuit of those who have been profiting from ransomware attacks, since we're seeing some evidence that it's having an effect.

In this case the suspect is named Denis Dubnikov. He was arrested two weeks ago, on the 2nd of November, while attempting to vacation in Mexico. Whoops. He was denied entry into the country pending an arrest warrant, and Mexican officials forwarded him to Amsterdam, where he was officially detained by Dutch police at the request of the FBI. Although arrests of crime suspects usually remain unreported until official charges are filed, in this case the news of his arrest leaked through himself, who revealed his fate in an Instagram story he posted on his now-deleted account while he was in custody in Mexico, according to screenshots posted on Russian Telegram channels.

The details providing his arrest had remained secret until just recently, with neither the Dutch police nor U.S. officials responding to multiple requests for comment. So there's still much that's not public. The Wall Street Journal obtained and reported upon an

extradition request which revealed that Dubnikov stands accused of money laundering. According to court documents, around $400,000 in cryptocurrency assets tied to a Ryuk ransom payment passed through one of Dubnikov's accounts in 2018. And it's unclear whether the 400K passed through his personal account or through two cryptocurrency platforms - one called Crypto Coyote, and the other is EggChange which Dubnikov, who's a Moscow businessman, founded previously. A Bloomberg article published on November 3rd, the day after his arrest, named EggChange as one of the multiple shady cryptocurrency exchanges that are headquartered in a Moscow office building that's been tied to cybercrime money laundering.

And the best news is that the arrest has triggered outrage within the Russian cryptocurrency community, and I think that's great. Several prominent figures are demanding an official response and condemnation of Dubnikov's arrest from the Russian government. I think it's great that something that happened three years ago is now being loudly dredged up and acted upon today, since you can just imagine all of the other crypto-weenies out there starting to worry about what evidence from their own pasts can be reverse engineered out of the public cryptocurrency blockchains to produce irrefutable evidence of their past misdeeds. As we noted last week, the money obtained from ransomware is seeming to be a little less free every day than it was before, you know, when it was just a spree that all these guys were off on.

Okay. So I propose the rhetorical question, how do you go about defrauding web-based advertisers? So you're a 41-year-old Russian national by the name of Aleksandr Zhukov. It occurs to you that advertisers pay websites who have their ads displayed to people who are presumably influenced to some degree by those ads, and who may even occasionally click on one to find out more. So you establish thousands of entirely fake public websites hosted by legitimate cloud service providers and contract with those advertisers to have their ads shown on those site's pages. And because video ads pay more, you focus upon having those hosted.

But no one's going to those bogus sites because they have crappy synthetic content. So you create a wide-ranging network of bots called "Methbot" to visit those bogus pages instead from IP addresses scattered across the world as if they were actual people. You design the bots to poke around the various website pages, to "view," in quotes, those ads and videos, and to occasionally click on one to learn more. And the money starts rolling in. How much money? Is everyone sitting down? Between $3 and $5 million per day at its peak in 2016. But of course it's fraud, since no one whose behavior might be influenced to purchase anything ever sees those ads for which advertisers are paying real money, and money is being received.

So something sets off some scam alarm somewhere which causes the FBI, Google, and 20 other tech and security firms to begin probing and dismantling, sort of unweaving Zhukov's operation. They soon discover a giant botnet running across infected devices and legitimate data centers spread around the world. And oddly, that botnet is visiting these bogus websites and clicking its links and ads. Shortly afterward, back in November of 2018 while traveling to Bulgaria, Zhukov is arrested on an FBI-issued warrant and is formally charged the next week.

Tracked under the names 3ve, as in Eve with the capital E backwards, Methbot, Boaxxe, and Miuref, Zhukov is believed to have run one of the largest ad fraud botnets ever created, generating, as I said, at one point in 2016 between $3 and $5 million in revenue per day. He hired programmers to help him build and manage his operation, which he disguised under the guise of a legitimate advertising network named "Media Methane." And according to prosecutors, Zhukov referred to himself frequently as the "king of fraud" and his employees as "my developers."

But of course, instead of running an actual advertising company, prosecutors said that Zhukov created thousands of fake websites where he loaded ads from legitimate publishers. These websites, they say, ran on more than 2,000 servers rented across data centers, and had no real visitors. Instead, Zhukov's team used automated tools to simulate real visitors. Prosecutors said Zhukov often leased IP addresses for this activity. And in other cases he hijacked legitimate IP addresses from their legitimate owners.

But all that came to a just end last week when a judge sentenced Zhukov, a U.S. judge sentenced Zhukov to 10 years in prison for running Methbot between 2014 and 2018. So yeah, you know, he probably should have quit while he was ahead, certainly he was at some point far ahead, and then just gone off to an island somewhere. But then I guess you can't spend all that money, if you're on an island. So I don't know what you do. You certainly don't want to travel. That seems to be when we get these guys.

I've got some really fun closing-the-loop feedback from our listeners. Marcio wrote, he said: "Hello. This may seem like a strange request in 2021." And Leo, you'll remember this. He said: "But would you be able to offer some insight on the exact SCSI commands used by your Click of Death tool Trouble in Paradise?" He says: "It might surprise you that anyone is still using Zip drives in 2021, but SCSI Iomega Zip drives are still used by people who collect vintage Apple Mac computers."

He says: "Obviously these computers can't run TIP," and for those who don't know, TIP stood for Trouble in Paradise, which was my Click of Death utility. I actually created it after the release of SpinRite 5 because SpinRite 5 was the first SpinRite that would run on Iomega Zip and Jaz drives. And those drives had a problem with their heads such that they would nick the edge of the soft flexible media inside of those Iomega cartridges. So people were using SpinRite 5 to try to recover the data. But because it was like such a weird kind of one-off problem, it quickly occurred to me, the moment I heard that we were getting reports saying that SpinRite fixed the Click of Death, and it's like, the what?

So I looked into the Click of Death, and I ended up writing a free utility which was much more problem-specific, whereas SpinRite was sort of a general solution. Because what I learned was you could, if you - the Click of Death could spread throughout a collection of Zip drives. If you inserted a Zip cartridge into a bad drive, it would nick the edge. But then if you then transported that cartridge, and of course transporting them was the whole point of having a cartridge, you then inserted it, this cartridge with the nicked Mylar, into a good Zip drive, it could damage the good Zip drive in such a way that that damaged Zip drive would then damage other Zip cartridges. It was literally a virus, like a mechanical virus, back in those days of Iomega Zip drives.

Anyway, he says: "Obviously these computers," meaning the Apple Macintoshes, "can't run TIP, but I recently came across a PC SCSI adapter card at a surplus sale, and I decided to see if I could get TIP to run on a Windows PC." He says: "It turned out to be an ordeal due to the fact that Windows 98 simply doesn't run very well on newer hardware. But eventually I got it all to work, and I am very impressed," he wrote, "with the tool," meaning TIP. He says: "I was quickly able to troubleshoot my collection of Zip media and drives." And that's of course why I created it. He says: "Anyhow, I would very much like to see if I could recreate a similar tool that runs natively on my vintage Macintosh computer so I can ditch the jerry-rigged Windows 98 PC and maybe help other Mac users who are also using Zip drives. Could you provide any insight on the SCSI commands used? Signed, Marcio."

And I zipped up, so to speak, my archive from whenever that was and mailed it to him. I got an email from him yesterday saying that he had made great progress. He was, you know, my assembly code is very easy to read. So although he's not an assembly code person, he is in the process of reimplementing TIP, Trouble in Paradise, on a native Mac. And it turns out I'd forgotten to include an include file that had my equates for the SCSI

commands that are unique to the Iomega, which I had somehow found, reverse engineered, or I don't remember now what. And so I sent that to him this morning. But anyway, just very cool, a little bit of a blast from the past.

Leo: No kidding. Wow.

Steve: Yeah.

Leo: That's a lot of effort to go through to use your old Zip drives, but okay.

Steve: Well, if you're a hobbyist...

Leo: It's fun.

Steve: And you're in love with old Macs...

Leo: You do it for the sake of it. Yeah, exactly, yeah.

Steve: Yeah. I got a neat tweet from a professor. His Twitter handle is @Prof_RAS6. Maybe that's, I don't know, like the designation for the course he teaches. Anyway, and Robert, I don't know how to pronounce your last name, Sciabbarrasi. Anyway, thank you for the tweet. And he tweeted: "Thanks for the dive into IRQs. I assigned that segment of the show as a reading/listening/watching assignment for my Computer Architecture class. Keep the deep dives coming." So Robert, thank you for a little bit of positive feedback.

And get a load of this, Leo. From a Bell Labs patent attorney, Stephen J. Phillips, whose email is still @bellsouth.net, the subject was Unix Programmer's Manual from Security Now! 844. And remember that we put the cover of it as our Picture of the Week, and that it was early November, maybe it was the 3rd, was the 50th Anniversary of at least that cover page.

Anyway, Stephen wrote: "The cover of the Unix manual brought back many memories for me. I was the Bell Labs Patent Attorney who consulted for the Mathematics Research Department where Ken and Dennis worked."

Leo: Wow. Neat.

Steve: Yeah. He said: "My bookshelf was lined with all the manuals as they issued. I spent many hours with the manuals and with Ken and Dennis, trying to find out what made Unix tick."

Leo: Wow.

Steve: And having worked with patent attorneys before, that is what a good patent attorney does. I mean, they're typically engineering trained. And so the inventor describes to the attorney, imparts the knowledge, explains what it is they invented, so that the attorney can properly create the legalese patent documents, which become indecipherable, unfortunately, but that's the nature of it.

Anyway, he said: "They were very modest about their work and thought they hadn't really done anything new. For the record, I filed two patents, now long expired, one for the Set User ID Bit which allowed a non-admin process to have administrative rights for the duration of its existence. The second was for the block-oriented Unix file system structure." He said: "I think our Patent Department was the first production user of Unix. We were looking for a word processing system, and they set us up with a PDP-11 doing text editing with ed. Later, we built a homegrown filing system for our patent applications and called it the File Mechanization project. Early days. We had fun after Unix became famous, when we were called into AT&T corporate headquarters to explain to the executive suite what in the world Bell Labs was doing creating software when telephones were our real business."

Leo: Wow. Really, really interesting.

Steve: So Stephen, thanks for the very cool blast from the past.

Leo: Yeah.

Steve: Ben Clapp tweeted: "Hey, Steve. Love the show. FYI, in case no other listeners have shared, Visual Studio Code is now rendering the Unicode directional formatting characters by default. I hope more IDEs do this." And Leo, this is the episode that you missed. But it's great to see this happening. I mentioned last week that GitHub was now going to be highlighting and spotting this. This is a perfect example I have in the show notes, where you can see an invocation of a function transfer balance, where it shows the from, to, and amount, and it shows 5678, 1234, and 6776 respectively as from, to, and amount. But the actual code down below, because there are hidden direction reversal Unicode characters, shows that the code that the system would process is different.

Leo: It's backwards.

Steve: Exactly.

Leo: It's the opposite. Holy cow.

Steve: Exactly.

Leo: Which is a lot more money, I presume.

Steve: Exactly.

Leo: Holy cow.

Steve: Certainly not what was intended because that Unicode 202E is the right-to-left override, and the Unicode 202C is the directional pop which pops the effect of the override. So Unicode itself turns out to be a stack-based interpreter. And this is what bad guys have actually gotten themselves up, I mean, some instances of this have been found in public source repositories. So this has actually happened.

Leo: Wow. Wow.

Steve: Very, very...

Leo: Clever, yeah.

Steve: Yeah. Oh, and a little bit of an update on SpinRite, which continues to - I'm the proud father. I'm nearly finished. No, it's turning out so nicely. I'm nearly finished with the 6th development release. The screen shot I've got in the show notes shows SpinRite's updated drive selection pane on the left, which now displays what most SpinRite users actually want to know, the estimated time to scan each of the drives upon which SpinRite's benchmark has been run on. And so you can see under this scan time column is the number of minutes, or in a couple cases hours, for really large multi-terabyte drives, how long SpinRite will take.

The first line item which I'm showing shows that the performance of a 250GB Crucial SATA 3 SSD under SpinRite is 573 megabytes per second. Okay, now, SATA 3 has a line rate, like a physical electrical bit rate of 6 gigabits per second. But SATA is a serial self-clocking protocol which requires some encoding of the user's raw data. For example, if you were to send all zeroes, which meant no bit transitions, then the self-clocking would fail because there'd be no way for the other end to keep track of precisely, you know, 6 gigabits, I didn't do the math, but that's 6x10 to the, what, mega is 6, giga is 9; right? 6x10^9. You flip that over, that's how short the bit window is. And you have to account for jitter and so forth in the line.

Anyway, the point is that it's necessary to encode the actual data being sent into something that's guaranteed to have a minimal number of bit events, no matter what data is being sent. Consequently, the 8 bits of data, every 8 bits of data is encoded into 10 bit symbols for transit over the wire. Therefore, that raw 6 gigabits per second on the wire translates to exactly, since every byte turns into 10 bits, 6 gigabits turns into 600 megabytes per second of possible data, 600 megabytes per second.

But a SATA serial link doesn't only carry the user's data. Right? I mean, there's no other channel. So all of the link's non-data control signaling also shares the same wire. So SpinRite's new hardware AHCI drivers are achieving on that Crucial 250GB SSD 573 megabytes per second out of, well, you can't get it any faster. The theoretical byte transfer is 600MB. We're at 573, which is 95.5% of that 600 megabytes per second. So we're only losing 4.5% to the control signaling overhead, and there's no way to squeeze that down any further. And given that most SATA 3 benchmarks you'll find show between 550 and 560 megabytes per second, I'm very pleased that SpinRite is achieving a sustained rate of 573 because it's literally, you know, it cannot go, SATA 3 cannot go any faster. Those little SATA 3 cables are smoking.

Anyway, since I think anyone who's interested in SpinRite might be curious to know more, I created a four-page PDF which shows a bunch of SpinRite's new screens, along with some editorial commentary about what the various drives and controllers are. It's our shortcut for the week. So this episode is 845, so the shortcut is grc.sc/845, for anyone who is interested. And anyway, just a nice little checkpoint on where we are. I'm just about finished with, as I said, the sixth development release. I can't wait to get it into the hands of its testers. It's performing better for me than anything has previously.

In fact, this one incorporates that whole new poly IRQ solution which I implemented after we learned conclusively that there was a BIOS that was lying about which IRQ line its controller hardware was using. So I thought, okay, that means I can't care. So now SpinRite doesn't, and I think that's going to make a big boost. I'll certainly have news about that next week because I'm sure I'm only a few days away from getting it into the hands of our testers.

Leo: I hand you over to the blacksmith himself, Mr. Steve Gibson.

Steve: Okay. So we have the return of Rowhammer. Probably the biggest and most significant security news of the week appeared yesterday with a report published by the COMSEC computer security group. As I noted at the top of the show, we've spoken of the work of these guys before, but only identifying them as being a team at ETH Zurich.

COMSEC is the computer security group of the Department of Information Technology and Electrical Engineering. Their primary research goal is the construction of reliable and secure computing systems so their research often touches on all layers of the computing stack, from software all the way down to hardware. They explain that they use novel analysis techniques to better - yeah, certainly in this case - to better understand the attack surface of modern systems and, when appropriate, build systems that can withstand different classes of attack. Their just-published work is their design of "Blacksmith," which represents the maturation of Rowhammer attacks.

Now, to briefly remind everyone, "Rowhammer" refers to a broad class of so-called bit-flipping attacks upon the unfortunately fundamentally flaky nature of today's dynamic RAM memories, you know, the big high-density DRAMs that we're all dependent upon. I'll crib from Wikipedia since it's perfectly summarized there. Wikipedia says: "Rowhammer is a security exploit that takes advantage of an unintended and undesirable side effect in dynamic random-access memory (DRAM) in which memory cells interact electrically between themselves by leaking their charges, possibly changing the contents of nearby memory rows that were not addressed in the original memory access. This circumvention of the isolation between DRAM memory cells results from the high cell density in modern DRAM and can be triggered by specially crafted memory access patterns that rapidly activate the same memory rows numerous times."

Now, we've been tracking this, like the problem, since its first appearance, believe it or not seven years ago in 2014, and it's been a constant background issue for the industry which never quite goes away. The same research group developed their "SMASH" attack at one point, which was a JavaScript-based attack that gives the attacker an arbitrary read and write primitive in a web browser. Then, Leo, you'll remember there was "Drammer," which was the deterministic Rowhammer attack on mobile platforms. And we had fun with the "Flip Feng Shui," which was hammering a needle in the software stack. And then there was an over-the-network "Throwhammer," due to its distance. And now we have "Blacksmith."

In other words, this has been fodder for lots of research. And again, as I said, we never really get this thing resolved. The reason we cannot seem to shake this is that the

problem the first Rowhammer research revealed is not, as we say, a bug of DRAM. It's unfortunately an unwanted feature. It cannot truly be fixed, because it has always been an essential truth incorporated into the way DRAM operates. And in that respect it's a little bit like all of the problems we had with Spectre and Meltdown and the optimizations that Intel had incorporated into their chips. It's like, it's in there. It's kind of baked. And we're hoping that it doesn't get really bad.

Now, historically there have been other noisy and usage-pattern-sensitive mass storage systems. Those real old-timers among us will remember that core memory technology which was used in the early mainframe and minicomputers of the 1960s and '70s were similarly sensitive to access patterns with adjacent bits. Memory diagnostics were run with things like so-called "checkerboard" patterns in an attempt to store and retrieve worst-case data patterns. If your core memory back then was acting up, you wanted to determine that while running a diagnostic, and not while processing real-world data that might trip the same flaw by mistake.

And even today the venerable MemTest 86, now MemTest86+, which is over at MemTest.org, remains a viable and worthwhile test of a system's memory. I've had occasion to run it through the years when something seems a bit flaky, and I want to rule out a problem with DRAM. DRAM is big. Processor access to DRAM is much slower. And something like MemTest must bypass the cache, you know, the Level 1, Level 2, and Level 3 cache. So it's surprising how slowly it runs when you're actually testing DRAM directly. But that's what a good memory test has to do. So it's the kind of thing you run overnight.

And in the case of DRAM's inherent vulnerability to Rowhammer attacks, the best we can do is attempt to minimize and mitigate what is unfortunately a pervasive threat. We've heard, we have been told with a great deal of hope and some expectation, that the newest generations of DRAM, DDR4, were going to resolve this problem by incorporating specific anti-Rowhammer mitigations. So how'd that work out? The short and depressing answer is that the DDR4 memory protections have been broken wide open by Blacksmith.

The COMSEC guys explain. They said: "We demonstrate that it is possible to trigger Rowhammer bit flips on all DRAM devices today with little effort, despite deployed mitigations on commodity off-the-shelf systems. This result has a significant impact on the system's security as DRAM devices in the wild cannot easily be fixed, and previous work showed real-world Rowhammer attacks are practical, for example, in the browser using JavaScript" - that was their own work - "on smartphones, across VMs in the cloud, and even over the 'Net.

"Rowhammer," they say, "is a vulnerability caused by leaking charges in DRAM cells that enables attackers to induce bit flips in DRAM memory. To stop Rowhammer, DRAM implements a mitigation known as TRR (Target Row Refresh). Our previous work showed that the new n-sided patterns can still trigger bit flips on 31% of today's PC-DDR4 devices. We propose a new, highly effective approach for crafting non-uniform, frequency-based Rowhammer access patterns that can bypass TRR from standard PCs. We implement these patterns in our Rowhammer fuzzer named Blacksmith and show that it can bypass Target Row Refresh mitigations on 100% of the PC-DDR4 DRAM devices in our test pool." They had 40. "Further, our work provides new insights on the deployed mitigations."

So how did they do it? They conducted a series of experiments beginning with the observation that all of the previously used Rowhammer attacks used uniform patterns such as single-sided, double-sided, and n-sided attacks. The patterns were uniform in both pattern and in number. So this made them curious about DRAM's sensitivity to non-uniform attack patterns. Since the search space of non-uniform attack patterns is huge,

they conducted a series of experiments to determine the structure of patterns that effectively bypass the Target Row Refresh mitigations; and they discovered that the order, regularity, and intensity of accessing aggressor rows - "aggressor rows" is the term now of art in Rowhammer attacks of rows that you read in order to cause your target row to get itself confused. So again, the order, regularity, and intensity of accessing aggressor rows in non-uniform patterns form an essential component of successful attacks.

They noted that their observations matched well-known parameters within the frequency domain, namely frequency, phase, and amplitude. So they used those frequency domain patterns to design frequency-based Rowhammer patterns that can effectively explore the space of non-uniform attack patterns. They implemented these patterns in their black-box fuzzer, which they named "Blacksmith." Blacksmith is able to determine suitable parameter values for any specific targeted device. So essentially it uses fuzz-based attacks to learn about a specific DRAM's sensitivities.

Okay. So how powerful and practical are the resulting attacks? They wrote: "For our evaluation we considered a test pool of 40 DDR4 devices covering three major manufacturers - Samsung, Micron, and SK Hynix - including four devices that did not report their manufacturer. We let our Blacksmith fuzzer run for 12 hours to assess its capability to find effective patterns. Thereafter we swept the best pattern, based on the number of total bit flips triggered, over a contiguous memory area of 256MB, and reported the number of bit flips caused. The results were that our Blacksmith fuzzer is able to trigger bit flips on all 40 DRAM devices with a large number of bit flips, especially on devices of manufacturers which they anonymized as A and D from their research.

"We also evaluated the exploitability of these bit flips based on three attacks from previous work: an attack targeting the page frame number of a page table entry to pivot it to an attacker-controlled page table page." Now, that's one of the things we talked about in the past, remember, that if you could arrange to flip a single control bit on a page table from read-only to read-write, the jig's up. You know, the OS absolutely depends upon its page tables being protected. All you have to do is flip that bit, and you have a massive attack. Also they did an attack on the RSA-2048 public key that allows recovering the associated private key to authenticate an SSH host successfully, and an attack on the password verification logic of the sudoers.so library that enables gaining root privileges. All attacks were successful.

They said: "Our work confirms that the DRAM vendors' claims about Rowhammer protections are false and lure you into a false sense of security. All currently deployed mitigations are insufficient to fully protect against Rowhammer attacks. Our novel patterns show that attackers can more easily exploit systems than was previously known."

Okay, so their work has been assigned a CVE number 2021-42114. And their paper, titled "Blacksmith: Scalable Rowhammering in the Frequency Domain" will be presented during the 43rd IEEE Symposium on Security and Privacy. I have a link to their 19-page PDF research in the show notes for anyone who's interested. And they assembled an FAQ to ask and answer some top-of-mind questions. First they asked: "Are there any DIMMs that are safe?" The answer: "We did not find any DIMMs that are completely safe. According to our data, some DIMMs" - are more dim. "Some DIMMs are more vulnerable to our new Rowhammer patterns than others.

"Which implications do these new results have" - or, well, they meant what implications - "do these new results have for me?" And answer: "Triggering bit flips has become more easy on current DDR4 devices, which facilitates attacks. As DRAM devices in the wild cannot be updated, they will remain vulnerable for their service life.

"How can I check whether my DRAM is vulnerable?" And this is interesting. "The code for our Blacksmith Rowhammer fuzzer, which you can use to assess your DRAM device for bit flips, is available on GitHub. We also have an early FPGA version of Blacksmith, and we're working with Google to fully integrate it into an open-source FPGA Rowhammer testing platform." That would be, okay, several things cool there. First of all, anybody who wants to compile their Rowhammer fuzzer from GitHub can. And I will make the following offer. If anyone does, I'd like to know. I imagine you have to boot a system. I don't know what OS platform this thing's hosted on because you'd want access to all of memory. But maybe they just satisfy themselves with running within, you know, like allocating as much memory as they can and then pounding away on that. We'll see.

The FPGA is interesting. That's a Field Programmable Gate Array, which means that it would be possible to create a piece of hardware into which you plug a questionable DRAM or any DRAM and press a button, and it goes to town fuzzing that DRAM in order to determine how vulnerable it is. Because there wasn't room for it in the show notes, but in their paper they show their results across all 40 of their tested DRAMs, their DDR4 DRAMs. And it is the case that some had four bit flips; some had 400 million bit flips.

In other words, there's a massive range of vulnerability, obviously by manufacturer. Some manufacturers are having a higher bit flip rate than others. And also by DRAMs within a manufacturer, probably within a production batch. I mean, it's based on the marginal-ness of the actual silicon that these things are being printed onto. So I expect we're going to be hearing a little bit more about all this in the future.

They ask: "Why hasn't JEDEC fixed this issue yet?" And they answer: "A very good question. By now we know, thanks to a better understanding, that solving Rowhammer is hard, but not impossible. We believe that there's a lot of bureaucracy involved inside JEDEC that makes it very difficult." So, "What if I have ECC-capable DIMMs?" They answer: "Previous work showed that due to the large number of bit flips in current DDR4 devices, error connection will not provide complete selection against Rowhammer, but makes exploitation more difficult." And we did talk about it in the past about the fact that you can bypass ECC by flipping a pair of bits rather than just one.

They ask: "What if my system runs with double refresh rate?" They answer: "Besides an increased performance overhead and power consumption, previous work showed that doubling the refresh rate is a weak solution not providing complete protection." They ask: "Why did you anonymize the name of the memory vendors?" They said: "We were forced to anonymize the DRAM vendors for our evaluation. If you are a researcher, please get in touch with us to receive more information." And, finally: "Does Blacksmith work on DIMMs from other vendors?" And they answer: "According to statistics, the three DRAM chip manufacturers considered by us cover 94% of the DRAM market. However, we have tested Blacksmith on three DRAM devices from another vendor, and we could successfully trigger bit flips on these devices, as well."

So this is not a story that has a happy ending or anything that we can update or patch, or even hope for a patch in December. But it is critically important research for the industry. We all understand that today's security is porous. And we've never before been in a climate that's placing more pressure against our porous security boundaries than we are now. Bad guys are going to weaponize this research. That's clear. That's what they do. The problem, as has been observed, is that the world is already filled with DDR3, very vulnerable, and DDR4, still vulnerable, DRAM, all of which has now been shown to be very much more vulnerable than we believed before, and indeed to be practically vulnerable, that is, these are not theoretical. You can actually get up to some mischief with this. So the consequences of these bit flipping attacks are serious.

And as I said, there's been talk of an FPGA, which would be very cool because it would create some hardware. But we know that there is a range of existing vulnerability from

"not so much" to "very." So, you know, if there was a practical solution for qualifying the vulnerability of DRAM, that's certainly not going to make the vendors very happy. But you can imagine people over-ordering, you know, get 12 when you need eight, and select the eight that are the least vulnerable and return the 12 because you didn't mean to order them in the first place. Return the four, I meant, that are the most vulnerable. Of course that creates a pool of higher vulnerability DRAM that other people will be purchasing. So there's no good solution.

Anyway, I think I'm curious enough to go take a little peek over in GitHub and see what's there. I'm not going to spend any time on it. SpinRite is all I'm doing right now, folks. But anyway, we've got a lot of other listeners who are obviously very talented and interested in security. So if any one of our listeners does something with the research on GitHub, definitely let me know. I'd like to know, and I'd like to share your results with everybody else here.

Leo: Yeah, cool. I guess, I don't know, if it's typical by brand, or if it's just that particular stick. I don't know. We'll see. Maybe a lot of people do it, yeah. That concludes this edition, the Blacksmith edition of Security Now!. Steve Gibson is at GRC.com. That's the place to go for the world's finest mass storage maintenance and recovery utility, SpinRite, currently 6.0; 6.1 is coming. And you can participate in the development, plus get a free copy when the upgrade comes, if you go right now and buy 6.0 at GRC.com.

You'll also find there 16Kb audio for this show, if you are bandwidth impaired; of course the full 64Kb, as well. And it's the only place you can get the transcripts which Steve commissions, and thank you for doing that, and thanks to Elaine Farris for writing those. GRC.com. Lots of other free stuff there, as well, worth checking out. And you can leave him feedback there at GRC.com/feedback. Steve's DMs are also open on Twitter, @SGgrc. So you can ask a question or send him a Picture of the Week there.

We have a copy of the show, both audio and video, at our website, TWiT.tv/sn. We also have a YouTube channel dedicated to Security Now! so you can watch it there. Of course you can subscribe in your favorite podcast client. That way you'll get it automatically the minute it's available. Those who want to watch us do it live. And there are people, believe it or not - talking to you, Grayson - who are listening live. And they do that at TWiT.tv/live. There's audio and video there. Chat live at irc.twit.tv or in our Club TWiT Discord. All of that is just for you. We thank you so much to our Club TWiT members for making it possible, and of course to all of you for listening. And most important, thanks to Steve Gibson. Have a great week, Steve. We'll see you next time on Security Now!.

Steve: Thanks, buddy.