The Data Transfer Project

Description: This week we examine still another new Spectre processor speculation attack, some news on DRAM hammering attacks and mitigations, the consequences of freely available malware source code, the reemergence of concern over DNS rebinding attacks, Venmo's very public transaction log, more Russian shenanigans, the emergence of flash botnets, Apple's continuing move of Chinese data to China, another (the fifth) Cisco secret backdoor found, an optional missing Windows patch from last week, and a bit of Firefox news and piece of errata. Then we look at "The Data Transfer Project" which, I think, marks a major step of maturity for our industry.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-673.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-673-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got a great show for you, including, my goodness, hell's freezing over. The moon has turned blue. Google, Facebook, Twitter, and Microsoft helping people move from one service to the other? Steve explains all, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 673, recorded Tuesday, July 24th, 2018: The Data Transfer Project.

It's time for Security Now!, the show where we talk about the latest security news from a guy who knows, who knows all and tells all, Steve Gibson. Hello, Steve.

Steve Gibson: Hey, Leo. Great to be back with you again. And you know, we're coming up on the end of year 12.

Leo: Holy moly.

Steve: I think our first podcast was in August of, well, nearly 12 years ago, but I don't remember which one. But I think that's always sort of the time it comes around. So here we are, Episode 673 for July 24th. So coming up on the end of year 12. It'll be fun to be in year 13. Okay. So until an announcement, which was...

Leo: First show was August 18th, 2005.

Steve: Ah, perfect.

Leo: Just checked the website. Number one. Yay.

Steve: Yeah. So, wow, less than a month to go, and we're in year 13. So I was going to talk about, believe it or not, still another new Spectre processor speculation attack.

Leo: No. No. Holy cow.

Steve: There's, yeah, like we said at the beginning of the year, our architecture is so wrapped up on using speculation in order to improve performance that getting out, like backing out of this is going to be a nightmare. But something more important happened that I thought, okay, I'm excited about this because it represents, I think, a major step in the maturation of our industry. And it's known as the Data Transfer Project. And believe it or not, I mean, I can't quite believe it. Facebook, Microsoft, Google, and Twitter, to name the four big ones, and with others, like, pending, have decided to agree on an open source trans-service data transfer standard, specifically to allow users to move themselves easily between services.

Leo: Wow.

Steve: I know. So it's like the reverse of the walled garden.

Leo: Yeah.

Steve: I mean, it's shocking. And it actually exists. I mean, there's...

Leo: Do you think it's GDPR that's kind of triggered this?

Steve: Yes, yes. There is a mention of GDPR-ness. But, and this is what we're going to talk about at the end of the podcast is, like, what their goals are, what they're trying to do. Of course there are obvious security and privacy concerns because this allows a person to - well, and also business concerns. I mean, this is what surprises me is that a company would be willing, in return for being able to have a user easily import themselves, also easily able to export themselves, to say, "So long, suckers, I don't want to do this anymore with you," and go somewhere else easily. But that's the point of this whole thing. So anyway, to me, this is a huge, as I said, a step of maturity for our industry that I want to talk about.

So we will talk about another new Spectre processor speculation attack. We've got some news on DRAM hammering, which we've been talking about for a number of years. The original hammerers have some news that I didn't talk about a couple weeks ago because it was like, okay, more of this? But also now some good news. So it's like, okay, let's talk about - we'll kind of catch up on that.

We have the consequences of freely available malware source code floating around the 'Net. The reemergence of concern over DNS rebinding attacks, which you and I talked about in Episode 260. But there's been a big furor about that recently so I wanted to

remind our listeners what that is and sort of revisit this. We have what I recognize now as, okay, I'm just not very hip, apparently.

Leo: Why ever would you say that, Steve?

Steve: The idea that Venmo users are deliberately publishing, in public, every transaction by default, along with their comments to their fellow transactee, just boggles my mind. I just can't get my head around that. It's like, what? Anyway, so there's an interesting link that I want to share with our listeners. And you can, like, see the photos of these people who are sending money back and forth. We've got a little bit more Russian shenanigans, following up on last week's, you know, we're "All Up in Their Business" podcast. The emergence of a new trend, which I just called "flash botnets" because that's what they are.

Also some news of Apple's continuing move of Chinese citizens' data to China, basically this week or in the last week finally giving over the last vestige of control to a Chinese-owned telecom provider. Believe it or not, another, which makes it the fifth, Cisco secret backdoor found in major Cisco software. We have an optional missing Windows patch from last week, optional for users of Windows 7, Server 2008 R2, Windows 8.1, and Server 2012 R2, so just a quick note about that. Some Firefox news, a bit of errata, and then we're going to take a look at this sort of amazing Data Transfer Project, which actually exists. So I think another good podcast two hours from now will be in the can. But right now the can is still open.

Leo: Shows how cynical I am, that instead of thinking, oh, these companies are doing the right thing for their customers, I'm assuming it's something like the General Data Protection Regulation from the EU that's forcing them to confront this issue. Maybe they're altruistic. I don't know. Maybe. It's possible.

Steve: It could happen.

Leo: Could happen.

Steve: It could happen.

Leo: Could have Microsoft, Google, Apple, yeah, it could happen. They're working for us.

Steve: Why not?

Leo: They care about us.

Steve: So believe it or not, we have the vulnerability that keeps on giving with Spectre. Researchers from the University of California at Riverside have published a paper detailing yet another brand new attack that affects Intel, AMD, and ARM speculative execution which - get this - bypasses all existing recent software and firmware mitigations against speculation execution attacks. That is, this is a new type of attack on

speculative execution which none of the stuff we've done this year fixes because - I know. It uses an entirely different mechanism in all modern architectures than what we've been focusing on, which is caching and branch prediction. It uses something known as the "return stack buffer." So consequently this is named SpectreRSB.

Now, okay. In podcasts for the last nearly 12 years we've often talked about the stack. It's been a mixed blessing, especially for security people, because it is in buffer overruns that the stack is often participating. And it's because the stack has traditionally been in executable memory that, if you loaded a buffer that was, as the phrase is, "on the stack," and could somehow arrange to get program execution to jump into the stack, that is, into the buffer that you had provided, and it was executable, you could run the code that you had remotely supplied. So we have a remote code execution situation.

More recently, because the stack is normally meant to store data and really isn't needed to store code, updates to architectures have marked the stack NX, nonexecutable. So these problems have largely gone away. In the early days of Windows it was interesting. The movement of bits on the screen, the so-called "bit blit" operations where you would move a rectangle on the screen as part of the windowing operations, in the very early days, before we had GPUs, Windows would actually write code to perform that operation on the stack and deliberately execute it.

Leo: So there was mass copy of data from one memory location to another; right?

Steve: Right, right. But they didn't want to have, I mean, they were so concerned about performance that they couldn't use general purpose algorithms with lots of "if" tests and loop counts and things because doing that would slow it down too much. So to do the operation they would actually build the code on the fly, stick it on the stack, and run it there in order to get the absolute maximum performance from this particular operation. So, and of course those days have long gone, but executable stack remained and has now finally disappeared because it was just such a security problem.

But the way the stack works is it's a sort of a general purpose scratchpad. And it comes into main use in two instances. When you're in some code, and you want execute a subroutine, that is, a block of code that performs some function, you normally do that by pushing the parameters for that function onto the stack. So you have some values, and you push them on the stack. You then jump to a certain location as a subroutine. And what that means is that, at the end, that subroutine needs to come back to the instruction after it was called. So that the jumping to a subroutine, what the processor does is it puts the return address on the stack.

So you first push the parameters to be used. Then, as you jump to the subroutine, the address of the instruction after that jump is pushed on the stack so that, when the subroutine has finished executing, it knows how to get back to where it was called from. So the stack has a combination of parameters and return addresses. And inside the subroutine it might need some scratchpad area. It might need what are normally called "local variables." So it's able to use the stack itself. It'll allocate some memory on the stack to contain its own little working memory area. And they're local because they don't need to be retained after the subroutine executes. So it's perfect to put them on the stack because they just get popped off the stack, and they sort of disappear.

So all of this works. But in thinking through every possible way to make that process faster, where the Intel engineers and in general processor engineers were like, okay, where else could we squeeze out a little more performance? They said, hey, rather than having this software stack which mixes all this other stuff, let's have a separate hardware stack where we don't store the calling parameters, and we don't store local values. It's

only for the return addresses, thus it's called the "return stack buffer," and it's in hardware. It exists, it's not very deep, that is, the buffer itself is like typically between four and 16 entries or return addresses deep. And it doesn't have to be very deep because maybe you're only going to a subroutine and then coming back, which would only need one memory. Or if that subroutine calls another subroutine, which then returns to the first subroutine, and the first one comes back to where it came from, that would require two and so forth. So it doesn't need to be very deep because it normally gets reused a lot.

But what this allows is when you're speculating, and you're wanting your processor to run ahead - I mean, after all, that's what this whole thing is, is it's prefetching instructions that may be executed. And it's doing as much work ahead of time as it can to keep the system as busy as possible. So when the system is speculating, and it hits a return instruction, it knows, when the processor actually gets around to doing that, it's going to be doing a return from the stack. But between then and now, the stack may go through a whole bunch of other things.

So the point is the speculator can't use the value on the actual software stack because it may be changed, like from popping some other stuff off of it, before the processor actually gets to the return instruction. Yet the speculator, which is wanting to run ahead of that, needs to know where to return to. So that's where the return stack buffer comes in. Since the return stack buffer only contains return addresses, the speculator can know with high confidence that when that return instruction is eventually executed, even if the software stack has pushed and popped and all kinds of things happen to it in the meantime, that return instruction will be taking it back to the location on the return stack buffer.

So the point is that, by the time the processor actually gets ready to do the return, the return stack buffer and the actual physical stack will be synchronized. But the processor can't wait for that because it wants to run ahead. So the return stack buffer provides sort of a pure, simplified, mini stack that allows speculation. And, not surprisingly, that can be leveraged. And that's what these UC Riverside guys have done. And it turns out it is a potent attack.

They said: "In this paper we introduce a new attack vector, Spectre-like attacks that are not prevented by deployed defenses," meaning nothing we've done so far this year helps. Because, again, this is a completely different, orthogonal from all of the other speculating systems, vulnerability. They said: "Specifically, the attacks exploit the return stack buffer (RSB) to cause speculative execution of the payload gadget that reads and exposes sensitive information. The RSB" - and then they explain it - "is a processor structure used to predict return addresses by pushing the return address from a call instruction on an internal hardware stack," they say, "typically of size 16 entries. When the return is encountered, the processor uses the top of the RSB to predict the return address to support speculation with very high accuracy. We show that the RSB can be easily manipulated by user code."

They say: "A call instruction causes a value to be pushed to the RSB, but the stack can be subsequently manipulated by the user so that the return address no longer matches the RSB." That is, it's easy to cause a deliberate mismatch which can then be detected because that will change timing.

They said: "We describe the behavior of RSB in more detail, showing an RSB-based attack that accomplishes the equivalent of Spectre Variant 1 through manipulation of the RSB instead of mistraining the branch predictor," which of course was all about what the first variant of Spectre was doing at the beginning of the year. "We use this scenario to explain the principles of the attack."

Anyway, what I liked about this - oh, and by the way, this also cuts through Intel's highest level of protection known as the SGX, the Software Guard Extensions. They demonstrate a successful attack on Intel's software guard extensions, which is Intel's Secure Enclave technology, just rendering it useless. They finish, saying: "Current systems are fundamentally insecure unless speculation is disabled. However, we believe that it is possible to design future generations of CPUs that retain speculation, but also close speculative leakage channels, for example, by keeping speculative data in separate CPU structures than committed data."

And this is, as I've been saying, this repeats the feeling that I've had as we've developed over the last seven months a more mature understanding of just how bad this is. I mean, where as I was talking about last week, maybe we end up having to just segregate processes on processors and not allow the fully free cross-hardware usage of threads and processors and VMs and all because there's just - speculation is too important. We can't turn it off or performance will collapse. Yet we can't have it on where threads are altering the processor's future as a consequence of its past, which is what speculation leverage does for us. So, I mean, we've really painted ourselves into a box until we rethink how to do all this.

Okay. So I titled this next thing "Rowhammer, RAMpage, and ZebRAM." So our friends from the Systems and Networking Security Group at VU Amsterdam have been, as we have covered through the years, making a series of troubling discoveries about the potent effects from pounding on dynamic RAM memory, which we first talked about as Rowhammer. And then there was Drammer and, like, a series of attacks.

Well, about a month ago there was news of yet another DRAM-based hammering attack which the news and their coverage said would affect all Android phones released since 2012. That was RAMpage. 2012 is when Google introduced the ION memory manager which was part of Android 4.0, the Ice Cream Sandwich version. So RAMpage came from these guys. And basically it was malware that gets into any Android device since Ice Cream Sandwich, since Android 4, which is basically all devices since 2012, can obtain sensitive information using DRAM-based hammering. And our listeners know that this is, basically, you allow software to repetitively read from DRAM. And as a consequence essentially of noise, you can occasionally get a bit to flip in a physically adjacent row, thus the original Rowhammer name for this.

So I have a link in the show notes to a non-Google Play Store Android app which these guys have updated. They created a Drammer app for Android to check for the vulnerability of their earlier Drammer attack. So what they've essentially done is they have matured this attack, and they've updated their app to identify devices vulnerable to this newer and more powerful RAMpage attack. I have the link in the show notes for anyone who's interested. You need to side load it, so get it and load it into your Android device, but not from the Google Play Store.

Now, they have also produced an open source mitigation known as GuardION, G-U-A-R-D-I-O-N. Of course ION is the memory manager, so GuardION. And I've got the link to that in the show notes. What they now have is a new technique they call ZebRAM, Z-E-B-R-A-M, which the good news is, is a mitigation. We don't yet know what it is. They will be delivering a paper at the upcoming OSDI, which is a USENIX conference being held at the beginning of October this year, south of me in Carlsbad, California. The paper, although there's no details on what this is beyond the title, which is ZebRAM, comprehensive and compatible software protection against Rowhammer attacks.

So that's good news, potentially, which it's nice to have some good news finally from these guys. They have thoroughly looked into how to create these attacks and, based on what little we know, it looks like there may be a strong mitigation, which if added to Android, either as a third-party add-on - but I can't imagine why Google wouldn't

incorporate it into Android if it works and it doesn't have any obvious downsides - may offer, as they say, comprehensive and compatible protection against DRAM hammering, which would be great. So I'm sure at the beginning of October we'll be talking about what this is because that would be good news.

We also have an instance of a very powerful Android banking trojan source appearing on the Internet. What appears to be the case is that the group that had created what has been regarded by the security community as the most powerful, because of the way it operates, Android banking trojan known as Exobot, it must be that the group decided to go do something else because they first began offering the source for sale, which is normally something that only happens when they've decided it's in their interest to leverage their investment in technology by sort of cashing out and then going to do something else. So after it was sold a number of times, one of the purchasers apparently decided, oh, what the heck, it's everywhere now, I'm just going to put it out on the Internet. So it became free. And the free source for what is arguably the most effective trojan on Android for stealing banking credentials represents a problem.

So the security firm ThreatFabric, who's been following this, has stated that it's an unusually potent banking trojan, capable of infecting even smartphones running the latest versions of Android, which is something that very few trojans can do. The spokesman and security researcher at ThreatFabric said that all threat actors have been working on timing injections, meaning various types of overlay attacks, to work on Android 7, 8, and 9. However, he said, Exobot is something new. Exobot gets the package name of the foreground app without requiring any additional permissions, which apparently is not something that is normally available. And he says it's a bit buggy, but it works in most cases.

So he says the interesting part is that no Android permissions are required, whereas all other Android banking trojan families are using the accessibility or the use stats permissions to achieve the same goal. But that means they require user interaction with the victim. Exobot does not. So what's expected, and we may be talking about this, is a result of the source code now being freely and widely available. Security researchers expect now to see a rise in Android malware based on this technology. And why not? Hackers who now have a source of better solutions for their own malware will immediately start to use it. So be interesting to see, unfortunately, interesting but worrisome to see what happens.

Okay. So DNS rebinding. A post over on Medium last month generated a lot of interest. It wasn't news, but it's sort of fallen off people's radar. Brannon Dorsey made the post on Medium.com. I have a link to his post titled "Attacking Private Networks from the Internet" - again, listen, "Attacking Private Networks from the Internet with DNS Rebinding." Lifehacker picked up on it, as did a whole bunch of other people. If you put "DNS rebinding" into Google right now, you get, like, everybody all scrambling around as if this was something new.

Well, this is not something new. We did a podcast on it, podcast #260. But it is of new concern, arguably, because in the past, well, this first appeared in 2008, so a decade ago. Dan Kaminsky, who of course made a name for himself over in the DNS world, he brought it to focus in 2010, which is when we talked about it on this podcast. And as a consequence of what's happened in the last eight years with the explosion in IoT, its significance does resurface, and the world has really not been paying it attention. So I am, first of all really glad for Brannon's rekindling of this and refocusing.

There is a company, Armis, who sells IoT security products at the enterprise level, who has a page on their site. As a consequence of their survey of their customers, their page is titled "DNS Rebinding Exposes Half a Billion" - with a "B" - "IoT Devices in the Enterprise." And I have in the show notes and, Leo, you have it on the screen, a snip, a

screenshot from their page where they show 87% of switches, routers, and access points are vulnerable, that's 14 million by their estimation; 78% of streaming media players (Apple, Google, Roku, Sonos, for example), 5.1 million globally; 77% of IP phones from all the regular players (Cisco, Dell, NEC, Polycom); 75% of IP cameras (the GoPro, Sony, Vivotek), and there's 160 million of those; 66%, two thirds of our printers, all of which are typically located on our Intranets, on our local networks (HP, Epson, Konica, Lexmark, Xerox), 165 million printers; and 57% of Smart TVs, those integrated with Roku, Samsung, Vizio, 28.1 million.

Okay. So what are rebinding attacks? What is all this? Well, I will read from GRC's operation page for the DNS Benchmark because I incorporated DNS rebinding attack awareness there. In the DNS Benchmark, GRC's DNS Benchmark, there's sort of a circular icon over on the left for every DNS server. And if you are protected from rebinding, you get sort of an extra moat around the inner dot which is meant to show you that you're protected.

So I said on the operation page: "The outer circle of the resolver status icon shows what, if any, DNS rebinding attack protection the corresponding DNS name server provides to its querying clients." And I'll mention that OpenDNS does provide protection, and I used them in the screenshot because they're like the only one I could find that was doing this.

"DNS Rebinding attacks," I write, "utilize DNS to fool a browser's scripting security into believing that local resources such as the user's own computer or router or now IP camera, Roku devices, printers, Sonos speakers, whatever IoT devices are located in the same web domain as the script's source." Okay, in other words - actually I just say: "When this occurs, the browser's same-origin policy protection is bypassed, giving scripts unrestricted access to the local resource. This allows scripts to do bad things such as change LAN router settings or access any resources and computers on the LAN." I said, in parentheses: "(That's not good.) Security-conscious DNS nameservers are able to help block these attacks simply by never returning IP addresses that fall within the ranges of IP addresses commonly used with private LAN networks, behind a router, or the localhost IP, 127.0.0.1, which computers use to refer to themselves.

"GRC's DNS Benchmark tests each nameserver to determine whether it blocks, that is, filters the return of these reserved private IP addresses in both IPv4 and IPv6 formats. At the time of this feature's release, only OpenDNS nameservers can be configured to do this, and then only for IPv4. IPv6 versions of these queries are still able to sneak through. Since there is never any reason to return a private IP address from a public DNS request, all nameservers should block the return of private IP addresses. Hopefully more will in the future."

Now, I need to add a caveat there because it appears that there are some instances where corporations are deliberately using public DNS to refer to resources on their own internal LANs, when those internal LANs are using nonroutable space like 10.172, 192.168 and so forth. So there are exceptions. However, those should probably be handled with whitelisting.

Okay. So anyway, I said at the bottom of this: "Note: If you would like to learn more about the consequences and prevention of DNS Rebinding attacks, this was the topic of our Security Now! podcast #260. During that episode Leo and I explained the problem and discussed all the details of this at some length." And I said: "The whole story is available for download," blah blah blah. That was Episode 260.

So, okay. So what's going on here? DNS Rebinding is a - we've talked at length about the same-origin policy in browsers, which is crucial, which has to be enforced, which browsers restrict what JavaScript can do to the origin, that is, the domain name from which that script was obtained by the browser. So essentially it creates sandboxes, sort

of dynamic sandboxes so that script running on your browser that came from domain XYZ, it's able to do other stuff with domain XYZ. It's able, for example, using Ajax, to subsequently perform HTML queries to get additional resources, to examine and set cookies, and do pretty much anything it wants with that XYZ domain, that is, the domain where it first came from.

But script is explicitly, JavaScript explicitly prevented from doing anything with any other domain. That's crucial. Otherwise it could just reach out and mess with other domains globally or in the browser, other browser tabs, other pieces of the same page. You know, an ad, JavaScript running in an ad could have access to the domain that is hosting the ad, which you absolutely don't want because this is untrusted script. We require that containment.

Okay. So here's the problem. A clever DNS server for a malicious ad could deliberately use a short expiration DNS response, that is, when this malicious server is asked for the IP of its domain, it gives it, you know, whatever it is, a valid public IP from which resources are obtained. But it gives it with a deliberately very short DNS expiration. We know about DNS caching, how it is caching that allows the DNS system to survive because IPs for domain names tend to be relatively static. If they change, it's typically very infrequently. This allows that knowledge to be cached out across DNS servers on the Internet.

But there is a time limit. There's a timeout. Often it's a day. Sometimes it's eight hours. It's a function of how often IP addresses are expected to change. Again, not very often. And the longer you allow your own DNS server's IPs to live, the lower the load on your server because the IP records are expiring less often. So the servers out on the Internet are having to come back and refresh their record of your domain's IP less often, which reduces the load on your DNS server. So there's a tradeoff.

But a malicious server could deliberately set a timeout to, say, a minute. Now, sometimes this just doesn't work because there are caching servers that deliberately ignore such short timeouts. Although it's not in the spec to do that, some of them do set a minimum timeout. But a lot of servers honor the timeout in the DNS record. So this malicious script running on your browser in a malicious ad first gets a valid IP for its domain. Thus that loads the malicious JavaScript into that window. It then begins performing additional queries to that domain. But because this was cleverly set up ahead of time, and there's lots of ways this could be done to make it robust, unfortunately, that domain name-to-IP mapping will expire, which will cause the DNS system to ask for an update, ask again for an IP.

Now imagine if 192.168.1.1, which is oftentimes the IP of your local router, if that IP is returned in response to additional queries by this malicious script. The browser doesn't know about IPs. The browser knows about domain names. So it doesn't know, realize, or care that now that domain name which the script has legitimate same-origin policy access to, is pointing to your router, 192.168.1.1. That now frees the JavaScript in your browser to talk to the router, to start guessing web admin usernames and passwords. And if you're somebody who - oh, and of course we know that it makes a query to the router. It immediately gets from the response headers the router's make and model and version and all kinds of other information that the router happily broadcasts in its response headers. Now it knows what type of router you have. So it says, oh, goody, and knows what the default username and password is.

The point is many users who are - even savvy users setting up a local network mistakenly assume that inside the network we're all one big happy family, and that you're not going to have something malicious on your LAN because, after all, it's in your house. It's in your environment. So imagine someone who doesn't take the trouble to change the internal web admin from their router's default. This provides a mechanism

which works today to allow script which you didn't write to get into your network with essentially breaking out of the same-origin policy using DNS rebinding, which I've just described, to have free and unfettered access to the inside web admin of your router, which as we also know allows it access to Universal Plug and Play. It can open up incoming ports statically and allow bad guys in. So this is all bad.

Okay. So where does IoT come in? Well, this is bad in an instance where you left the web admin authentication set to its default so that the script could see what make and model router you had and then guess the default, or just sit in the background making other guesses, trying to brute force. But the big problem here, the thing that everyone's talking about now, what has changed in the last eight years is the prevalence of IoT devices and their implicit assumption that anybody on the local network is friendly. And Leo, I know you're a Sonos user, as am I.

Leo: Yes, yes.

Steve: And one of the coolest things about Sonos is that you don't have to do any weird login username or password stuff. The other day, actually Lorrie lost her previous iPhone, so she had it insured, and she got a replacement. But she said, "Hey, I want to control our music. What do I have to do?" And I said, "Nothing." So we downloaded Sonos, opened it. It came up and said, "Looking for your system." And it said, "Found." And that was it. No authentication.

Leo: Right, there's no authentication. Right. I'll vouch for that.

Steve: Which is - yeah. I mean, and it's wonderful. I mean, from a user ease-of-use standpoint, it's incredibly cool. But it happens that the downside of this ease of use is - now, you might argue there's nothing bad you could do with Sonos. It actually turns out Sonos allows you to run some Linux commands through its HTTP interface.

Leo: Uh-uh. Oh, dear.

Steve: So it can be used as a pivot for them establishing a foothold in your LAN and working from there. Sonos, by the way, just in the last month, Sonos and Roku, both of whom have had no authentication of their internal LAN API, have been reawakened to this problem as a consequence of Brannon's posting and have announced updates. And in fact I did get a notice on our Sonos that there was a new version available, and I imagine they've decided they need to close some of these holes because, while it's convenient, this really is a problem.

So for the last eight years, since my DNS Benchmark was published, I've been running a "rebindtest" server. If people who are a little tech-savvy are interested, we all have typically an nslookup command. In Windows, there's nslookup, and I think it's called the same thing on Linux and Unix systems. If you do "nslookup net192.rebindtest.com," that's my DNS server that I've been running for eight years in order to support GRC's DNS Benchmark that does rebind testing just as part of the service that it offers. You can do net192, net4, net10, net127, net172, or net192, any of those .rebindtest.com. What you will almost certainly see, unless you are an OpenDNS user and you've turned on rebind protection, you will get a local IP back. For example, net192.rebindtest.com will return the IP 192.168.0.1.

Leo: It did.

Steve: Yes. And just to make sure, net4 will return 4.4.4.4, which is a valid public IP as we know. That's one of - I think it's one of Level 3's DNS servers. I just did that in order to verify that some public IPs are coming through and private ones are not, in order to prevent false positive rebinding notifications in the Benchmark. But the point is there's no good reason for a public DNS server, which rebindtest.com is, to return the IP of an internal LAN, 192.168.0.1. Yet it does. If you are configured under OpenDNS, that will not happen, so you'll know you're protected. So a simple way of checking is just to manually do an nslookup using my rebindtest service, or you can just use the DNS Benchmark if you're a Windows user, and it'll show you. And I did it this morning. There were none. Not a single DNS server out there is doing this.

And in that screenshot, though, I think I do show one where there's like a circle, like a semicircle, looks like it's maybe, what, the sixth one down, is it OpenDNS? I think it ends in 202.202? I can't seen the screenshot from there. But yes, you can see that - yes, that's the one, 220.220. And so that almost enclosed outer ring, the top segment is 127.0.0.1. They're not blocking that. But they are blocking the other three RFC 1912 networks because in this case I had configured, for the sake of testing, I had configured OpenDNS to do that.

So there are some SOHO routers, looks like Asus may be among them, and I have an ASUS router - I'm using pfSense here in my workspace, in my work area, but I have an ASUS router at home. So I'm going to check it to see if it's got DNS filtering feature. There are some that are beginning to do that. And users can protect themselves from this and then verify using the Benchmark or just doing an nslookup, see whether you're able to obtain an IP address for your own internal network or any private network from a public server. This is a problem, and this is not something that we are currently protected from.

And so yay to Brannon for bringing this up, and to Sonos. Oh, also Google. Google's Home systems have similar default, no authentication required. Apparently he had to poke them a lot by showing them example after example after example. And on his page he does have some testing. It requires that your network be 192.168.1.x. But he actually has a testing page with JavaScript which will successively attempt to probe your LAN from a page running JavaScript and look for Google Home, Roku, and Sonos devices and tell you when it has found them because they are visible to script running in a web page. So a little bit of a wakeup call for, yes, it's so convenient not to have to authenticate our IoT devices. But it does require that everything be trusted on our LAN. And this is an instance where that assumption of trust can be broken.

So as I said at the top of the show, Leo, I'm bemused by the idea that all Venmo transaction details - now, not all details of the transactions, but some surprising ones, are all being maintained in a publicly accessible log.

Leo: Yeah.

Steve: And so if you click on this link, the first link I have in the show notes here, that just shows - it says "limit=1." That'll just bring up one transaction. And if I click on it, let's see, I'm seeing [Miss Tina Lashi] sent money to [Maya Zalastone]. And now...

Leo: I'm getting a different one, of course.

Steve: Yes. And if you refresh, you'll get a different one, and if you refresh, a different one. Because, I mean, there's a lot of transactions happening on Venmo.

Leo: Oh, yeah. It's huge.

Steve: And actually, in Firefox actually it formats the page nicely so that I see...

Leo: Oh, see, I'm not getting that. Safari's not formatting it because it's a, I don't know, it's XML, yeah.

Steve: It is. And so, for example, I mean, I've got picture links to these people. And if I click on a picture link, I see them.

Leo: Well, it's easier than that. If you just install Venmo it's right on the front page. That's why we know people know about this. But I should point out that I don't know if everybody knows about it because it is the default. And people might just be assuming that there's some privacy to this. So we've brought this up several times. See, this is on Venmo. This is the public feed.

Steve: Correct. And I think, in thinking about this more, I think it's one thing to say, oh, look, you know, like here's the public scroll. But I wonder if people understand that these transactions are memorialized in perpetuity. There was a researcher who is a privacy advocate who was curious about this. So he pulled all 207,984,218 transactions which occurred in 2017 from the public API. I mean, it's just there. I mean, so it's not a scroll that sort of goes away. Your entire transaction history in Venmo is there forever as publicly available.

Leo: The default is public, visible to everyone on the Internet. I immediately, when I started using Venmo, set it to private, and that means it's not in that database. It's not...

Steve: Logged.

Leo: ...viewable. Unless there's a flaw in Venmo's system. But I presume that's not what we're talking about. It's just that their defaults are public.

Steve: Yeah, exactly. It's just that, I mean, in thinking about it, again, maybe I'm just an old fogy.

Leo: People treat it like a social media feed. And they have fun with it because usually they say, you know, "illegal stuff" or, you know. I mean, they make fun with it.

Steve: Right, right. So anyway...

Leo: You should know about it, though. It is a potential problem. I mean, people might be paying for stuff and not want anybody to know it, and everybody does.

Steve: Yeah. And in fact this one guy who pulled all of the 2017 public transactions, that is, nearly 208 million of them, he created kind of a fun page: publicbydefault.fyi. I guess he's a funny Chinese guy, Hang Do Thi Duc is his name. But from looking at the data, he just pulled some random ones out. He tracked the transactions related to a cannabis reseller, a corn dealer, a particular family, a few random couples, and the story of a woman with 2033 Venmo transactions.

Leo: Wow.

Steve: So she's a big Venmo user. And you can put comments, as you commented, on the transactions. And they're all there for the world to see.

Leo: They tend to be emojis. People really like to use emojis.

Steve: Well, actually...

Leo: That tells me that they kind of know what they're doing.

Steve: In the links I was seeing, I was seeing people, like that Tina what's her name. She's got big glasses on and - yeah.

Leo: Well, yeah. You have your picture. That's what I'm saying. It's like social. So if you download the Venmo app, and you go back here to - let me see. Home and then the public stuff. I always use emojis. There are a lot of emojis in there. But this is the person. So it's just like a Twitter thing; right?

Steve: Yeah.

Leo: But, you know, some of it is silly. Some of it is intentionally kind of ribbing people. But I do think most people know it's public. It's just I worry about people who don't because...

Steve: Right. And people are, like, sending money to each other; right?

Leo: Yeah.

Steve: That's what it's for.

Leo: Yeah. I mean, I use it. I send money to my daughter. I use it to tip my massage therapist. I use it to tip my manicurist or pay my manicurist, stuff like that. A lot of people - this is a PayPal service.

Steve: Right.

Leo: A lot of businesses now are using Venmo. Originally it was used by millennials to share meal bills and stuff like that, or to reimburse for gas or rent or that kind of thing. So I think millennials kind of know what's going on.

Steve: It is worth noting that the transactions can be canceled within some length of time. I ran across that when I was trying to dig into what in the world this thing is.

Leo: I use it. It's a great service. I mean, there's others. There's Cash.me from Square. There's Zelle, which is the banks are doing that. But Venmo's probably the biggest. And we've actually talked about this for some time on iOS Today because people should know, turn that off.

Steve: Good, yes.

Leo: And you can do it on - you can also do it on a transaction by transaction basis. On every transaction there's a public/private button.

Steve: Right.

Leo: But I think people go very fast, and they're not paying attention. Yeah.

Steve: So anyway, a little bit of a public service and just a head scratcher for me. It's like, okay, I don't understand.

Leo: You're just too old, Steve.

Steve: I am. I'm an old fogy. So a real quick note that the Russians are - I have here the Russians are (still) coming, because of course you and I...

Leo: Great movie.

Steve: ...are old enough, and half of our listeners probably, to remember the movie "The Russians Are Coming."

Leo: Oh, yeah. Oh, yeah.

Steve: It was a comedy. So in this case not such a comedy. So following our topic, of course, from last week about the DoJ's indictment of those 12 Russian agents, I just wanted to note that during last week's very interesting for many reasons Aspen Security Forum, Andrea Mitchell had an interesting interview at length. But in this case Tom Burt, who's Microsoft's vice president for customer security, said that earlier this year they had discovered a fake Microsoft domain that had been established by Russia as the landing page for phishing attacks. Microsoft said it detected and helped the U.S. government to block Russian hacking attempts against at least three congressional candidates this year.

Although Microsoft declined to name the targets, they said the three candidates were "people who, because of their positions, might have been interesting targets from an espionage standpoint, as well as an election disruption standpoint." And according to Microsoft, the Russian hackers targeted the candidates' staffers with phishing attacks, redirecting them to a fake Microsoft website in an attempt to steal their credentials. And Tom Burt, who is this VP at Microsoft, said that they "discovered these fake domains were being registered by an activity group that at Microsoft," he said, "we call Strontium," which the rest of us know as Fancy Bear or APT 28. So it's that same group.

So for what it's worth, this continues. And I expect it will continue to continue. I hope that - and I didn't really say this clearly enough last week. After everything that has happened nearly two years ago, staff in elections, where they are inherently targets, really have to be trained up on security. I mean, the way this happens is phishing attacks. They are social engineering attacks of various sorts. And they just have to be trained up in order to just avoid the temptation of clicking on things in email. I mean, it's almost worth imagining filtering people's email to redirect the clicks to a local catch basin or just redact them from incoming email. I mean, just, like, not have any because they're just too dangerous. And people just, you know, what we see over and over is that people will get caught out by that.

Okay. We're also seeing something that I've termed here "the emergence of Flash Botnets." There was some transaction in Twitter. A researcher, Ankit Anubhav, I'll just go with that, Ankit Anubhav...

Leo: You're as good as I am on this one.

Steve: Boy. So in his Twitter feed he noted that an IoT hacker identifying himself as "Anarchy" has claimed to hack more than 18,000 Huawei routers in 24 hours using the vulnerability from last year, 2017-17215, which was used last Christmas, or leaked last Christmas, and was used several times by the Satori botnet. So he's taking responsibility for what was observed as a massive uptick in Huawei scanning, which several security firms had observed. In his dialogue his motives were not clear. But the attacker who Ankit corresponded with said he's making the biggest, baddest botnet in town. So maybe for DDoS.

Leo: Bad, bad botnet Brown.

Steve: Biggest baddest botnet, yeah. And unfortunately, and what Ankit said in his tweet was it's painfully hilarious - I don't think it's hilarious at all, I think it's just painful - he says how attackers can now construct big botnet armies using known, I mean, like well-known vulnerabilities. So anyway, so the 17215 was the well-known exploit. It used a Universal Plug and Play access to the WAN side, obviously, of Huawei routers. It had been abused by at least two versions of the Satori botnet and also by the Brickerbot and a number of smaller Mirai-based offshoots that we've also talked about. So it's just sort

of like there. And there's 18,000 of these, and nobody cares. It's now been, what, at least nine months, eight or nine months since this has been publicly known.

And again, we talked about this. Even though Huawei has been responsible, they are responsible for the problem, but mistakes can happen. They have an announcement on their site. They have taken responsibility. They've got updated firmware. But they don't have the ability to push that to their routers. Their routers are not checking periodically and notifying their owners or maybe even going the next step and maintaining their own security autonomously. I think we're at the point now, I mean, we've got - nobody has to update their Chrome browsers anymore. Nobody has to update Firefox. These browsers just take care of themselves. Same for Edge and IE. And as we know, there is some interaction at the OS level.

But at the appliance level it doesn't make sense. There ought to be something, I mean, nobody reads a license agreement anyway. It certainly doesn't say in there that your router probably has known vulnerabilities that allow an attacker to take it over remotely and get inside and roam around your network. Yet that's the reality. So there's no reason it couldn't say your router may, at a period of time of low usage, like 4:00 a.m., update itself with new firmware all by itself. I mean, you can have a checkbox that you have to turn off in the configuration if you want to explicitly disable that. But it ought to be on by default. We need to move to that place. In which case it would be possible for all of these Huawei routers to be phoning home, discovering, oh, I've got new firmware, update themselves, flush out as a consequence of rebooting anything that might have crawled into their RAM in the meantime, and patched themselves.

We have to be doing that for these Internet-facing appliances. It's time. I mean, the consequence of not doing it is what we're seeing, which is long-term, never going to go away, now known persistent vulnerabilities that can be endlessly abused. And now we have the ability to form, as we've just seen, an 18,000-strong router botnet within a day. Oh, and this guy says, by the way, he's not done. The Anarchy guy who did this using the Huawei vulnerability said he plans to next target an older vulnerability from 2014, that's 8361, which is a well-known vulnerability in Realtek routers, exploitable via port 52869. That still exists, and he's going to go and add those routers to his botnet next.

Oh, and in an update in some reporting of all of this in Bleeping Computer, they noted that both Rapid7 and GreyNoise are confirming scans for Realtek have gone through the roof today. So the guy was, yes, he followed through on his plan to go after Realtek routers next. So the only way that we're going to get out of this mode is ultimately for these existing routers to be replaced. But what they are replaced with has to then maintain itself. It just has to in order to connect to the Internet.

I did want to mention just sort of in passing that we did see in the last week the formal final turnover of Apple's iCloud data to a Chinese state-owned ISP and cloud service provider that is going to be managing it from now on. And so I just wanted to remind our listeners, it's not clear if you change your iCloud storage setting whether - and maybe you know from working with Rene and the guys over on MacBreak Weekly - whether that pulls your existing iCloud data back to the U.S., or is it only when you're setting up a new iCloud account and you opt to have your storage in the U.S.? Because that's an option that Apple provides. But the default would be for Chinese users of Apple iOS products, that their iCloud stuff would now be stored in China.

Leo: I don't know.

Steve: Yeah. So anyway, I did want to tell our listeners that there is that option. Don't know if it will delete what's in China and start using it in the U.S., or even if it can be

changed after the fact. But for what it's worth, I regard what Apple has had to do as inevitable. I mean, they want to operate in China. They want the Chinese market. The Chinese government has the right to decide by law what its citizens' access to data is.

So I don't see any choice here. I mean, we watched Russia fight with Telegram, and Telegram lose, because ultimately a local government can control what happens within its borders. So I don't see this, I mean, I know that civil rights people and privacy advocates are all screaming; but it's like, well, sorry, that's China. So Apple wants the Chinese market. At least users setting up iCloud for new devices can choose to have their data located in the U.S. I don't know how long that's going to last and what that really means. But it is an option.

Cisco has discovered and removed another undocumented backdoor. Last week Cisco released 25 security updates, among them a patch they rated with a severity of eight, I'm sorry, of 9.8 out of 10. So I don't know what it has to be to be 10. I guess maybe a wide-open, unauthenticated admin with no password on the WAN side of their products. That would definitely be a 10 out of 10. This got a 9.8 because what they discovered was another undocumented root admin password. This was one for the Cisco Policy Suite. And what the patch did was remove an undocumented password which had been built into the root account of all previous Cisco Policy Suite software. So it was a previously secret backdoor. They tracked it with a CVE-2018-0375. And it's significant due to the location of these systems within enterprises.

The Cisco Policy Suite has three editions - Mobile, WiFi, and Broadband Network Gateways which Cisco sells to ISPs and large corporate clients, allowing network admins to set up bandwidth usage policies and subscription plans for customers and employees by tracking individual users, the traffic tier, and to enforce access policies. So Cisco said there's no workarounds or mitigating factors, that customers need to install the patch to remove the secret password. They say they discovered the undocumented root password during internal security audits and believe that it may have been left behind during software debugging tests, as they say most of these incidents end up being.

But being the fifth such secret backdoor to be found, for me this is a headshaker, as I said when it was the third, and when it was the fourth, and now it's the fifth. This really no longer seems like a mistake. This seems more like a previous policy. And as I said before, I'm impressed by the way Cisco is behaving now while apparently under new management because they're saying, whoops, we want to remove these. And remember, someone had suggested, well, maybe this was in a product they acquired from somewhere else. So you could hold them blameless for stuff being in a product that they purchased and they didn't know about, like they own Linksys, for example. Who knows? But in any event I am impressed that they are saying, whoops, we found another one. At some point, though, when you get to five, it's hard to say, oh, well, yeah, this is another one got left in by mistake. It's like, okay.

Bleeping Computer noted that there was a patch missing from users of Windows 7 and 8.1, both in the workstation versions of those OSes and also in the server versions, meaning Server 2008 R2 and 2012 R2. It's not super critical, but it's not a patch that we will receive automatically. So I just wanted to put it on people's radar. Because I do run Windows Server 2008 R2, and this does bear on servers, I will definitely be applying it myself manually. So I've got the links in the show notes for anyone who is interested. For Windows 7 it's KB4345459, and for Windows 8.1 it's KB4345424. So again, KB43454 and then ending in 59 or in 24.

And so it fixes three things. It says: "Addresses the issue in which some devices may experience a stop error when you run network monitoring workloads; addresses the issue that may cause the restart of the SQL Server service to fail with the error 'TCP port is already in use'" - so that would be a problem, if SQL Server couldn't rebind to its

expected port. And then, finally, "addresses an issue that occurs when an administrator tries to stop the World Wide Web Publishing Service. The W3 Service remains in a 'stopping' state, but cannot fully stop or cannot be restarted." Well, I do that from time to time in order to update the code running on the - the GRC net engine code that I've written for the server. So that matters to me.

So anyway, I just wanted to put it on people's radar. Thanks to Bleeping Computer for putting it on my radar that those patches - oh, and for some reason, what's interesting is they noted Windows 10 did receive these as part of last week's monthly rollup. But 7 and 8.1 did not. So don't know why.

Lastly, as I had mentioned before, Firefox will be getting the autoplaying video with sound suppression, thank god, on web pages, although I think one of my add-ons, probably uBlock Origin, is able to block that for me. I go to sometimes use somebody else's machine or browser and stuff's, like, talking. And it's like, what? What? Anyway, I know, Leo, that's been a bugaboo of yours recently.

Leo: Oh, I hate it, yeah, yeah.

Steve: So Chrome and Edge are doing it. It's looking like Firefox 63, due near the end of October, will be getting the same thing. So yay. It'll just be the default is "Always Ask." And that's a little popup that comes in up under the URL, like sometimes I see them if I go to a local movie theater site, they want to know my location so that they can select the proper theater for me if they're a large chain. And so there's like a popup you get in Firefox. So this would be one that would ask whether you want videos that are wanting to autoplay, whether you want to allow them to or not.

So starting with Firefox 63 it is default to "Always Ask." But if you know you never want them to autoplay, you can change that setting to "Block Autoplay." Or if you know that you always want them to autoplay because you don't have enough noise in your life, you could change it to "Always Allow" autoplay. So thank goodness that Firefox will be getting that, too. I'm still using Firefox, just because I love the tabs on the side. I just, you know, if Chrome would fix that, you know, there are some add-ons that have tried to do that, but none of them integrate as nicely as they do for Firefox for me.

I have a piece of errata thanks to Herzi was his Twitter handle, I think maybe his name, also, H-E-R-Z-I. He said: "In SN-671 [so that's two weeks ago] you imply that the PortaPow [that's that cute little red] USB condom is a dumb USB plug that only forwards the power and ground lines." He says: "It does a lot more than that as it contains its own little logic board." And so I thought, what?

And so I dug into it, and it turns out that, yes, indeed. In the features for that cool little PortaPow it says: "SmartCharge - Built-in chip detects the type of device which is connected and swaps between Apple, Universal, and Samsung charging specifications. This prevents the blocker from slowing down charging and can increase charging speed if your charger is sending the wrong signal, for example, charging an iPad from a charger which uses the Universal charging spec."

So in fact it sort of - it effectively enhances the thing that you have plugged it into. And so, for example, maybe if you were using this in a car, and so you were just using a USB charger that was unaware of the iPad protocol, this essentially upgrades the charging port to one that is smarter. And then it did say, under limitations: "This adapter is not compatible with extra fast charging technologies such as the Qualcomm Quick Charge, the Samsung Adaptive Fast Charge," oh, and a Samsung...

Leo: Samsung. Samsung.

Steve: Samsung, thank you, Adaptive Fast Charge, "as these require data transfer to be enabled." That's interesting. "Your device will still charge at a high speed without these. Some sat navs and dash cams use a proprietary signal and will still try to enter sync mode unless their own charger is used. Some car USB sockets do not provide enough power to charge a device, so a dedicated USB car charger must be used." So anyway, Herzi, thank you for that, and I appreciate being able to bring all the proper information to our users.

Leo: Nice.

Steve: Okay. So the Data Transfer Project. As I said at the top of the show, I think this represents a major step forward in the maturation of our industry. Major players are participating - I'm just stunned by this - Facebook, Google, Microsoft, and Twitter. They explain that the project was formed in 2017 to create an open source service-to-service data portability platform so that all individuals across the web could easily move their data between online service providers whenever they want.

Leo: It's wild.

Steve: It is. It's incredible. And the URL is DataTransferProject.dev, Leo, if you want to bring it up while I'm explaining.

Leo: All right.

Steve: And it's the first link in the show notes. "The contributors to the Data Transfer Project believe portability and interoperability are central to innovation. Making it easier for individuals to choose among services facilitates competition, empowers individuals to try new services" - okay, now get this. "Making it easier" - I'm reading from their page. "Making it easier for individuals to choose among services facilitates competition, empowers individuals to try new services, and enables them to choose the offering that best suits their needs."

So under "What is the Data Transfer Project?" they explain: "The Data Transfer Project" - and I've edited this down for size and added some clarification in various places. "The Data Transfer Project is a collaboration of organizations committed to building a common framework with open source code that can connect any two online service providers, enabling a seamless, direct, user-initiated portability of data between the two platforms." Just I'm speechless.

"The individuals," they say, "should be able to easily transfer their files and data directly between online service providers. The Data Transfer Project extends data portability beyond downloading a copy of your data" - this is where you were talking about the GDPR before - "beyond downloading a copy of your data from your service provider to providing customers the ability to directly transfer data in and out of any participating provider. The project is an open source initiative to encourage participation of as many providers as possible. DTP will enhance the data portability ecosystem by reducing the infrastructure burden on both service providers and users, which should in turn increase the number of services offering portability."

What they mean by that, I'll just say that they recognize that users may have themselves a bandwidth limitation. And over time you could accumulate, for example, a bazillion photos. And it's just not feasible for you to download, if you wanted to move from a photo management facility, from one to another, they're literally making it possible to do that. They're saying, well, due to bandwidth constraints, it may not be technically feasible or possible for an individual to download all of their photos to hold them for then upload to another service.

So what they're talking about doing is creating a direct point-to-point interoperability among every pair of partners involved so that, given proper authentication, and that's where our podcast and our discussion of this doubtless in the future comes in, an individual would be able to securely authenticate and authorize the direct transfer of all of their photos from one of these participating services to another, and their bandwidth would not get tied up. They would say, "Yes, please, do this." And then the system would arrange to make sure they're who they say they are and authenticate them securely and would then do a direct inter-provider transfer so that new photo sharing provider number two receives the entire photo history from provider number one. It's incredible.

Leo: It's great. This is very much like Google Takeout, but with the cooperation of other services. It's brilliant. I can't believe they're doing this.

Steve: I know. It's mindboggling.

Leo: Yeah.

Steve: So they say - I'm scratching my head because it actually itches. But it's like, yeah. "The protocols and methodology, they say, of DTP enable direct service-to-service data transfer with streamlined engineering work."

So getting into a little more detail, how does it work? "The Data Transfer Project uses services' existing APIs and authorization mechanisms to access data. It then uses service-specific adapters" - that's what they're calling them, and I'll explain that in a second - "to transfer that data into a common format." So this project is coming up with a set of common things that people transfer, and then creating a vendor-neutral format into which each of them can translate in and translate out of. And so they say: "It then uses service-specific adapters to transfer that data into a common format and then back into the new service's API."

So this comprises three main components. They have data models which are "canonical formats that establish a common understanding of how to transfer data. Adapters provide a method for converting each provider's proprietary data and authentication formats into a form that is usable by the system. Task Management Library," they call it, TML, "provides the plumbing to power the system. Data models represent the data when being transferred between two different companies." So that's this common transfer format. "Ideally, each company would use interoperable APIs to allow data to flow between them. However, in many cases that is not the case. In those cases there needs to be a way to transfer the data from one company's representation to another company's representation."

So they say that "Data models are clustered together, typically by industry grouping, to form verticals. A provider could have data in one or more verticals." So verticals could be, for example, photos, email, contacts, or music. Wow. "Each vertical has its own set of data models that enable seamless transfer of the relevant file types. For example, the

music vertical could have data models for music, playlists, and videos. Ideally, a vertical will have a small number of well-defined and widely adopted data models. In such a situation, the generally accepted standard will be used as the data model for that vertical across companies." They say: "This is not currently the case for most verticals because data models have emerged organically in a largely disconnected ecosystem.

"One goal of DTP" - and I think this is really amazing, and this will probably happen - "is to encourage organizations to use common data models in their systems, which will happen if organizations take importing and exporting data into consideration when initially designing their systems or providing updates. Using a common data model will significantly reduce the need for companies to maintain and update proprietary APIs."

Okay. So what they're saying is this effort will be producing a set of sort of generic and universal specifications which can be used for inter-provider transport of this stuff. But if a new company comes along, rather than it reinventing the wheel and just using their own ad hoc system, it would be, if they - first of all, they're going to want, if they're a new service, they're going to want to always support DTP because that gives them the opportunity to acquire users because this creates user portability, against all walled garden logic. So it makes sense that they would natively support these existing models and not have to create an ad hoc system and then an interface layer. Why not just do it from the start? So it's like, yes.

Leo: Yeah. It seems like the right thing.

Steve: Yeah. And so they also have what they call "company-specific adapters," which you would have if you had already existing ad hoc stuff. They said: "There are two main kinds of adapters: data adapters and authentication adapters. These adapters exist outside of a provider's core infrastructure and can be written either by the provider itself, or by third parties that would like to enable data transfer to or from a provider." In other words, they're saying that a company might not themselves support DTP. But if they do have their own APIs that are publicly accessible, like Venmo, then somebody could come along and create - a third-party could create a set of adapters to interface that proprietary company's APIs to the DTP system.

So they say: "Data adapters are pieces of code that translate a given provider's APIs into these common data models used by DTP. Data adapters come in pairs: an exporter that translates from the provider's API into the data model, and an importer that translates from the data model back into the provider's API." And then there are authentication adapters, which are "pieces of code that allow consumers to authenticate their accounts before transferring data out of or into another provider." And this system will leverage OAuth. They said it's "likely to be the choice for most providers; however, DTP is agnostic to the type of authentication."

And then finally that last thing. I mentioned the TMLs, the Task Management Libraries. They say: "The rest is just plumbing. The Task Management Libraries handle background tasks, such as calls between the two relevant adapters, secure data storage, retry logic, rate limiting, pagination management, failure handling, and individual notifications. DTP has developed a collection of Task Management Libraries as a reference implementation for how to utilize the adapters to transfer data between two providers. If preferred, providers can choose to write their own implementation of the Task Management Libraries that utilize the data models and adapters of DTP." And I should mention that the third link in - oh, the second link is an overview whitepaper, I think it's 24 pages, which provides a lot more depth of information. And the third link at the top of this is all of this stuff on GitHub. It's all open source.

So they provide, just to kind of help people pick themselves up off the floor, some use cases. They say: "Individuals have many reasons to transfer data, but we want to highlight a few examples that demonstrate the additional value of service-to-service portability." So, first: "A user discovers a new photo printing service offering beautiful and innovative photo book formats, but their photos are stored in their social media account. With the Data Transfer Project, they could visit a website or app offered by the photo printing service and initiate a transfer directly from their social media platform to the photo book service." Meaning them not needing to be an intermediary, but saying to the social media platform, "Please send these photos to the photo printing service." And now those two services would know how to interoperate.

Another example: "A user doesn't agree with the privacy policy of their music service. They want to stop using it immediately, but don't want to lose the playlists they've created. Using this open-source software, they could use the export functionality of the original provider to save a copy of their playlists to the cloud. This enables them to import their existing lists to a new provider, or multiple providers, once they decide on a new service."

Or: "A large company is getting requests from customers who would like to import data from a legacy provider who is going out of business. The legacy provider has limited options for letting customers move their data. The large company writes an adapter for the legacy provider's APIs that permits users to transfer data to their service, also benefiting other providers that handle the same data type." So in other words the going-out-of-business provider doesn't have their own support for this common data model, but somebody creates one because it's in their interest to do so, and makes it available.

Or: "A user in a low bandwidth area has been working with an architect on drawings and graphics for a new house. At the end of the project, they both want to transfer all the files from a shared storage system to the user's cloud storage drive," in other words, directly into the cloud. "They go to the cloud storage Data Transfer Project user interface and move hundreds of large files directly, without straining their own bandwidth."

And, lastly: "An industry association for supermarkets wants to allow customers to transfer their loyalty card data from one member grocer to another, so they can get coupons based on buying habits between stores. The association would do this by hosting an industry-specific host platform of DTP."

So they explain, finally, that "The innovation in each of these examples lies behind the scenes. Data Transfer Project makes it easy for providers to allow their customers to interact with their data in ways their customers would expect." Well, in the future, wow. "In most cases, the direct data transfer experience will be branded and managed by the receiving provider, and the customer wouldn't need to see DTP branding or infrastructure at all." So that suggests that, once this is in place, it would be a get-data-from mode, where you would go to the - you'd be setting up a new service somewhere, and they would offer import from other services capabilities, just as other services would offer import from them. So it would be mutual and, again, just sort of stunning.

And they conclude with: "Why do we need DTP? Users should be in control of their data on the web. Part of this is the ability to move their data. Currently users can download a copy of their data from most services, but that is only half the battle in terms of moving their data. DTP aims" - I guess a typo - "aims to make moving data between providers significantly easier for users." So, wow.

Leo: Now, are these companies actually doing it? Twitter, Facebook, Windows, and Google?

Steve: Yeah. It is now in place. They've been working on it since last year, and this was the wraps coming off of the project.

Leo: Wow. That's really awesome.

Steve: Yeah. Like I said, it just feels like a huge step in maturity for our industry.

Leo: Yeah. I'm still wondering.

Steve: I know. It's like, what?

Leo: What are they up to? What are they up to?

Steve: So let's hold our breath and hope it works and hope that the security is locked down. I will certainly be taking a closer look at that as this actually surfaces, and we see how this works.

Leo: Well, you know, Google's always had this. This has been a been a big part of the Google services for a long time. They call it, what is it, data freedom.

Steve: Yeah.

Leo: And they've always believed in it. But there was no interoperability. It was just like a, you know, XML file or CSV file, and that's that. Which is better than nothing. I mean, I love that. But now they're going even farther, which is great. It's amazing.

Steve: Yay.

Leo: Yay. Wow, Steve. A show that ends on an upbeat note.

Steve: Yay.

Leo: Wow. Wow. Steve Gibson is at GRC.com. You've already heard he's got quite a few little useful tools there you might want to check out. Of course the most useful of which he didn't even mention, although somebody in the chatroom said, "I've got a SpinRite story for you." Let me scroll back and see if I can...

Steve: Oh.

Leo: Yeah, yeah. It was just a, you know, was a short little thing. Oh, I wish I could find it. Here's a SpinRite - this is Aneroid. "Here's a SpinRite testimonial. It's keeping

two of my 12-year-old hard drives alive, 500GB SATA 1, now in a one-year-old gaming laptop." Wow.

Steve: Nice.

Leo: Twelve years. That's almost as long as this show's been around. That's awesome.

Steve: Very nice.

Leo: Get your copy. GRC.com. SpinRite is the best data transfer hard drive - not transfer, data transfer, that's what we were talking about - hard drive recovery and maintenance utility.

Steve: Maintenance, yup.

Leo: While you're there get all sorts of stuff. SQRL. When are we going to do our SQRL thing? Now, you lost Father Robert unless you do this in August. He's going to be back one last time.

Steve: So I'm in the process still of bringing myself up to speed on a system I know nothing about, which is the web forums which I chose to host SQRL. We have to have SQRL log into its own web forums, but it's in minified JavaScript, which is impenetrable.

Leo: Right.

Steve: And PHP written on top of the Zend Model View Controller platform.

Leo: Oh, my god.

Steve: And so it's a little opaque, and it's proprietary, and it doesn't support doing what I want to do. So anyway, that's where I am. As soon as we get that done, then we'll be able to point everyone at it and have someplace for people to go with their SQRL questions because I just need to have public web forums available for handling the launch traffic.

Leo: Nice.

Steve: Getting there.

Leo: GRC.com. You can read all about it, find out what SQRL is. You can also get all sorts of other cool stuff. While you're there you might want to pick up a copy of the show. He's got audio, and it's the only place you can get transcripts of the show.

Elaine Farris writes those, and they'll be out a few days after the show comes out. We have audio and video at our website, TWiT.tv/sn.

And of course, as with all the shows that we do, you can get them on your favorite podcast application, Pocket Casts or iTunes or Google Podcasts or Stitcher or Slacker or whatever, whatever you use. In fact, all of those home assistants now you can just say, you know, "Listen to Security Now!," and maybe you have to say - sometimes you have to say "Listen to Security Now! podcast." I think on the Amazon you might have to say "Listen to Security Now! on TuneIn." But just try saying "Listen to Security Now!," see what happens. You'll probably get it, the most recent episode, and you can just listen. Walk around the house, do your household chores while you're listening. That's a good thing.

And we do the show every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. So you can tune in and watch it live. If you do that, visit the chatroom, irc.twit.tv. Next week I am not going to be here. I'm running out. Jason Howell will be filling in for me.

Steve: Are you having fun somewhere?

Leo: Yeah, we're going up to Tahoe for a few days with the kid.

Steve: Nice. Nice. Summertime.

Leo: Summertime. Enjoy some summertime fun, go river rafting and stuff like that. But I will be back the week following.

Steve: Jason will hold down the fort in the meantime.

Leo: He does a great job.

Steve: And now I've got my Skype audio settings working, so we're good to go. Woohoo.

Leo: Your sync is good and everything.

Steve: Yup.

Leo: Thanks, Steve. Have a great day. We'll see you next week on Security Now!. Bye-bye.

Steve: Thanks, buddy.