
# Sony Sues Quad9

**Description:** This week fewer questions required longer answers. What, if anything, can be done about the constant appearance of malicious Chrome extensions? What's the latest country to decide to pull Chinese telecommunications equipment from their country? What's the number one way that bad guys penetrate networks, and how has that changed in the past year? What delicate and brittle crypto requirement is responsible for protecting nearly $1 trillion dollars in cryptocurrency and TLS connections, and how can we trust it? What's now known about the Plex Media Server defect that indirectly triggered the exodus from LastPass? And why in the world would Sony Entertainment Germany bring a lawsuit against the innocent nonprofit do-gooder Quad9 DNS provider? Stay tuned! The answers to questions you didn't even know you had will be provided during this March 14th "Pi Day" 914th episode of Security Now!.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-914.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-914-lq.mp3

---

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to talk about that delicate and brittle crypto technology, the elliptic curve technology that has allowed hundreds of bitcoin wallets to be drained of their value; the latest in the Plex Media Server defect that caused the LastPass hack; and Sony's lawsuit against Quad9, all over an Evanescence album? It's next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 914, recorded Tuesday, March 14th, 2023: Sony Sues Quad9.

It's time for Security Now!, the show where we cover the latest news in the world of security with this guy right here, our master of ceremonies, Steve Gibson. Happy Pi Day, Steve.

**Steve Gibson:** Happy Pi Day to you, Leo. And I was saying before the show that we need a Pi Day on a Friday, but we don't have that this time. So this week we've got fewer questions which required longer answers. And so we're going to answer some questions. What, if anything, can be done about the constant appearance of malicious Chrome extensions? What's the latest country to decide to pull Chinese telecommunications equipment out?

What's the number one way that bad guys penetrate networks, and how has that changed in the past year? What delicate and brittle crypto requirement is responsible for protecting nearly $1 trillion in cryptocurrency and TLS connections, and how can we trust it? What's now known about the Plex Media Server defect that indirectly triggered the exodus from LastPass? And why in the world would Sony Entertainment Germany bring a

lawsuit against the innocent nonprofit do-gooder Quad9 DNS provider? Well, stay tuned. The answers to questions you didn't even know you had will be provided during this March 14th, Pi Day, 914th episode of Security Now!.

**Leo:** How many decimals of Pi can you do off the top of your head?

**Steve:** We had a serious geek in high school who used to just bore us with, you know, he would just...

**Leo:** Go on and on.

**Steve:** And of course we didn't know if he was right or wrong; right? It's like, okay, Richard.

**Leo:** Yeah, he's at some point just saying - I can do, because there's a mnemonic, I think George Gamow the physicist came up with it, how I want a drink, alcoholic of course, after the heavy chapters on quantum mechanics. So 3.1415926, of course after, 536, no, 7, Q-U-A-N-T-U-M, 7, mechanics, whatever that is.

**Steve:** So I used to - once upon a time I could rattle it off for like a while, but not...

**Leo:** I thought you could, yeah, yeah.

**Steve:** Yeah, that was a thing.

**Leo:** On Pi Day in the grade schools here, when Abby was a kid, some years ago, they would have a memorize pi thing. And they'd have a competition to see who could memorize the most digits of pi. Just crazy.

**Steve:** I was pretty good with, what was that thing that went boop and beep, and the pattern got longer every time?

**Leo:** Morse code?

**Steve:** Simon. Remember?

**Leo:** Oh, Simon, yes. Oh, wow.

**Steve:** Milton Bradley's Simon.

**Leo:** That was a tough one, yeah.

**Steve:** And the advantage was that you kept reinforcing what you already knew, and it only extended by one beep each time.

**Leo:** Then they came out with - and you probably would only know about this if you had kids - something called Bop It that was like a Simon, but you had to do different motions and rotate it around. And same thing, you had to memorize the order, and it would tell you what to do. And then if you got it wrong it'd go, "Oh." I'm sure there are people in our audience that remember that. I only know because, again, this is something the kids were into. Do we have a picture? I didn't even look yet.

**Steve:** Yes, we do.

**Leo:** Going to see it for the first time right now. Oh, god. Oh, lord. Set us up. What are we going to see here?

**Steve:** Okay. So my caption for this is "This was their solution?" And so this - and it looks kind of hazy and grainy because this is one of our listeners took a picture of the screen and tweeted it to me. He said, "Oh, this has got to be a Picture of the Week." And oh, indeed.

So this is you're logging in, and you get to the one-time password prompt for your login. And there's a field where it says "Enter OTP" in red. But above that, apparently they're having some sort of problem. So it says: "We are facing an SMS issue. Please use 910296 as your one-time password." And, oh, okay. So what occurred to me was, okay, first of all, this is weird; you know? But why not just disable the one-time password challenge for, like, while there's an SMS issue, whatever that could be, rather than hard-coding the proper answer for the one-time password field? Anyway...

**Leo:** It's not a phishing, couldn't be a phishing thing; right? I mean, it wouldn't be bad guys doing that.

**Steve:** No, I can't see how this could do anything except just be like, well, use this number because we'll accept it. Now, what would really be interesting would be to remember that number and see if it still works.

**Leo:** Forever. Forever.

**Steve:** After you've got this problem, yeah, like after they've got this problem solved. Maybe that's the backdoor, and they thought, well, we'll change it. In fact, we'll see an instance later in the podcast where there's a concern over people having put something in during testing which they forgot to remove later. But okay.

So first, after seeing the details of another Chrome extension that was seriously compromising the privacy and security of more than 160,000 Chrome users, you know, stepping back from this, it feels as though we need a better solution than we have currently, and I don't know what that would be. But we have a need for assuring what extensions are doing.

Okay. So in this case, the extension was called - and okay, it was a little sketchy to begin with, I guess - "Get Cookies.txt" was actually the name of the extension. And it was being offered through the official Chrome Web Store so it had been vetted and approved. It allowed users to export from Chrome the content of their cookie files in the old Netscape browser cookie format, presumably so that you had a copy of it, a record of it, for maybe importing it somewhere else.

Okay. But concerns about the extension were brought to light first of all a couple months ago in January when a Reddit user discovered that, beyond just allowing people to export their cookies from Chrome, and you do that once, right, and then remove the extension. Well, apparently people weren't.

This Reddit user discovered that the extension was tracking users by collecting user and browser data and uploading it on the fly to a remote server. At the time the Reddit user posted: "Version 1.5.0 of the 'Get Cookies.txt' extension is sending details of every page you visit, not just video sites, but every page, back to its developer at the domain ck.getcookiestxt.com." He said: "Specifically, for every page you visit, it sends a unique ID for your browser installation, your browser's user-agent string which shows what OS you're using and the browser version number, your language setting, the platform you're on, the current date/time and your current time zone, as well as the URL that you're visiting."

So that's not good, the URL you're visiting with a unique ID tracking you, by an extension that has no business whatsoever doing any of that. But then it later came to light that, after the version was changed, the extension was not only performing that bit of tracking that had been seen in January, but that it was also proactively sending entire user cookie files that you'd given it permission to acquire in the first place, sending the entire cookie files back to the extension's publisher. In an update to that Reddit user's initial posting last week, they wrote: "The situation is now even worse. The extension is now also sending all your cookies to the developer, too."

When that was confirmed, "Get Cookies.txt" was immediately pulled from the store. But that wasn't until the extension's upgrade has been in place for some time; and many users, we don't know how many of the 160,000 who were using it had obtained the update and probably had their entire cookie files, you know, all of their Chrome cookies sent to the extension's developer. And of course, after all, what are we constantly telling everyone they need to do? Stay current with all updates. So update to the latest "Get Cookies.txt." And unfortunately, that hurt you rather than helped you in this case.

So as we know, anytime we either explicitly enable the "keep me logged in on this machine" checkbox, or anytime a website chooses to do that for us for whatever reason, you know, on our behalf, this logged-on persistence, which is a convenience, significantly, since now we're using the web more and more for apps and things, that convenience is accomplished by causing the web browser to accept and store a long-life persistent cookie in our browser's cache. That cookie identifies us to the site and has the effect of keeping us logged on. Because individual browser events are separate transactions, the cookie accompanies each one of those queries to the site which says, oh, yeah, that's Steve again.

So anyway, since this is exactly the data that the "Get Cookies.txt" extension was caught sending home to its publisher, the publisher was obtaining the static session data needed to impersonate all of the users of its extension at any of the sites where a persistent session cookie may have been set. Which is increasingly what's being done because, as I said, we're using browsers more and more as apps. And since the cookie file indicates its expiration date, if any, in the future, it's trivial for the attacker to determine where you're currently logged in. That is, to separate the cookies which are just transient session cookies and persistent cookies being used to permanently log you in.

So for what it's worth, if by any chance you or someone you know as one of that extension's 160,000 users, you should seriously consider taking the time to log out of any websites where having some unknown bad guy logging in as you could be a problem. If you just log yourself out, that will render any of those stolen cookies that may have been exfiltrated from you useless.

And this takes us back to the question I posed earlier. Like this is happening all the time, you know, variations on this theme. How do we solve this sort of serious problem? We want our benign browser extensions to be both powerful and capable. Otherwise they're not that much use to us. So they need to know how and be able to do dangerous things, like have access to our browser global cookie cache.

But how can we safely trust extensions from unknown authors which might have a hidden agenda? I mean, obviously we're putting some trust in the vetting process, which presumably occurred in order for this thing to originally get itself listed. And even though it had a somewhat dubious purpose, even on day one, the fact that it was then able to change what it did, like it earned some transient trust for being there for a while, it also earned 160,000 users.

And you might be thinking that if the extension's developer was always - if their intention was always nefarious, then they may have well been biding their time, waiting for the total user count to get up to a point where they would then incrementally change what this thing did in order to get additional information on their user base until they finally said, okay, it's time for us to cash out and grab everybody's cookie files and see what mischief we can get up to.

So today's browser ecosystem doesn't really provide any mechanism for deeply vetting an extensions operation. I mean, that would require more effort than is obviously being put in and may be available for free extensions created by unknown people in who knows where. It would be prohibitively expensive to be able to fully examine every extension in detail. And to do that you'd really have to provide source code, which some extension authors might feel was putting a big of a chill on the whole idea.

**Leo:** They're in JavaScript, though; aren't they? Or can they be a binary blob?

**Steve:** There was some requirement that was added after this became a real problem that the extensions not be obfuscated, like not be minimized in order to deliberately make them smaller and to cram out all of the text. And they couldn't be encrypted. But still, I mean, you'd have to, like...

**Leo:** You'd have to deminimize it. You'd have to know how to unobfuscate it.

**Steve:** Exactly. And read through every line of this thing. And it's sort of clever, too, because, I mean, if this was always the guy's goal, then he first came up with something that needed the permission to have access to the cookie cache and had a valid reason for wanting that. So that would have meant that there was all of this code in there that was...

**Leo:** Legit, yeah.

**Steve:** ...reading and parsing and working with cookies. So somebody who casually looked over the script would go, well, yeah, okay, it's doing cookie stuff. So, fine.

**Leo:** Right. Extensions are hazardous. I mean, that's what Tavis Ormandy was saying when he said don't use your password extension. And now of course we know Bitwarden has an issue with its extension because it gets tricked by iframes. I suspect that's a problem with many password extensions.

**Steve:** Right, right. So interesting...

**Leo:** It's better to just use the app.

**Steve:** Interesting problem. And, yes, exactly. If you can, I mean, but at the same time look at all the - look at how much now we're using.

**Leo:** And you've got to cut and paste; right?

**Steve:** Yeah.

**Leo:** Which can be problematic.

**Steve:** And also look at how much we're using our browsers for, like...

**Leo:** We live in them, yeah.

**Steve:** Yes.

**Leo:** I have a ton of browser extensions, including UBlock Origin. We've talked about that.

**Steve:** Yup, yup. In fact, I've got one I love. It's a session save and restore for Firefox. I'll use it in order to move all of my work in progress while I'm working on the podcast on Monday here, I like save all of the tabs that I've got in a cute little JSON file and send it over to my other location where I'm able to open essentially exactly the same state that I was in before.

**Leo:** I guess remove any extension you don't need and you're not sure of.

**Steve:** Yup, yup. And here again, for example, would it have been those users' persistent need to export their cookies? It seems to me that's a one-time thing. I don't know why you would want to. But install the extension, use it to export your cookies, and remove the extension.

**Leo:** Yes.

**Steve:** Clearly 160,000 people just left it there because, oh, you know, it's not hurting me. Well, except it was.

Okay. So following the U.S., Australia, Canada, New Zealand, Sweden, and Britain, according to reports in German media, the German government is planning to ban the use of Huawei and ZTE equipment from their national 5G telecommunications network. German officials cited the increasingly common fears that Huawei and ZTE equipment could be used for Chinese sabotage and espionage. The German government had previously given the green light for Chinese equipment to be used for its 5G network, and some has already been installed. But the recent Russian invasion of Ukraine, of all things, and Russia's attempt to blackmail Germany from aiding Ukraine through its Nord Stream natural gas pipeline, led to a major change of thinking in Berlin. And the reporting was that those were the factors, which seems odd. Russia's not China. But apparently the two are a little too close for comfort.

According to German media, telcos that previously installed 5G technology from the two vendors may be forced to rip out and replace the equipment. China is understandably unhappy and said that they hope Germany will make their decision one way or the other without any outside interference, meaning primarily from the U.S. trying to be persuasive here. The collection of countries are already on various schedules to remove any existing Huawei and ZTE equipment from their networks.

And I continue to be distressed because, you know, this seems xenophobic given the lack of concrete evidence. But the argument always is there could very well be no evidence despite equipment being compromised. And congressional expert testimony confirmed that it is virtually impossible to know one way or the other. The good news is that the world does appear to finally be waking up to the broad potential for the abuse of today's advanced technologies. I'm hoping that what we're seeing may be just a bit of a reflexive reaction to this new awareness.

We've all had it, right, for the last 18 years of this podcast, knowing what was possible. Back in the early days of viruses, Leo, we were often just sort of musing over the fact that all they really seemed to do was propagate. They didn't really do anything bad. But we all knew they could. And every so often one would be a low-level formatter or something, which would definitely ruin your day. So I hope that this is going to calm down over time, and that we'll end up reaching some balance point which is sane.

The abbreviation DFIR is a term in the security industry, stands for Digital Forensics and Incident Response. And there is an annual DFIR report which is released, obviously annually, that came out last week. The report contains a bunch of interesting statistics. But probably the most interesting fact, because it's like maybe the most significant fact, was its breakdown of the way bad guys are still gaining their initial entry into our systems. And I've got in the show notes a big pie chart which is hugely dominated by one big slice of pie. In fact, it's not a slice of pie. It's the pie left over after you take out the slices of pie.

Here it's an overwhelming sea of blue which represents phishing. The way bad guys are still getting in, holding a 69.2% share of intrusion methods, where a victim was induced to click on a link or open a document which in every case was received via email, either through a mass mailing to an organization's employees, hoping that one of them will click on it to get in, or sometimes targeted to a specific individual in order to make the email more like something that they're expecting to see. So nearly, just shy of 70% of the way the forensic studies are showing people are getting in is somebody on the inside clicking on a link, inviting the bad guys in.

The DFIR report noted that as in 2021, the previous reporting period, also now last year, 2022, the majority of intrusions originated from mass email campaigns spreading malware in this way. They also noted, and I thought this was very interesting, that one of the biggest shifts in this space was the discontinuation of macro-based attacks leveraging Word and Excel, which was the result of that decision that we talked about at the time last year of Microsoft finally choosing to disable the default execution of macros for Word and Excel. I mean, all they had to do, and finally did, was pop up a warning notice, rather than just having them run by default.

Think about how long that has been the default case and for how many years before Microsoft did not take this decision their users, Windows users, were being hurt by this widespread use of default execution of macros in Word and Excel. Throughout the years before this we kept lamenting Microsoft, how they left scripting enabled in email, despite the fact that no one wanted scripting in email. And only the bad guys were abusing that. That battle to finally disable scripting was won, and now at long last we have, you know, another change was made that this report demonstrates markedly changed the nature of the attacks. So, boy. It just, again, one of the things we have seen in so many different ways in this podcast is the nature of inertia. Inertia is amazing in all of its many forms.

Taking second place in the "intrusion entry methods" at 15.4% is drive-by compromises. So those are the watering hole attacks where a malicious website is first set up, and then victims are lured to visit. Or in some cases just opportunistic, where a malicious site is set up, and somebody who jumps in compromises themselves. So that's 15.4%. The remaining two major ways of gaining entry into networks, which completes our pie, actually they share the remaining 15.4% split in half, each getting 7.7%. Public-facing applications is one of those by which they actually meant the exploitation of Microsoft Exchange when you drill down into the report further. And the abuse of valid accounts is the other 7.7%, meaning brute force remote desktop protocol or SQL Server entry in order to get onto a network. So that rounds out all 100%.

But that big blue slice of pie representing nearly 70% of all intrusions, and arguably preventable, it would take some effort, but it could be done. And we could hope that at some point somebody at Microsoft will wake up one morning and say something like, hey, you know, email phishing attacks are a big problem. Why don't we, I don't know, run email in its own sandboxed virtual machine so that bad stuff from the outside is contained and can't take over the entire machine? What do you think? Well, maybe in another decade we'll get email in a protected sandbox virtual machine.

Okay, now, Leo, this next one is a biggie. So I think we need to take a break.

**Leo:** Fair enough.

**Steve:** And then we're going to plow into something known as the "Polynonce Attack."

**Leo:** Uh-oh.

**Steve:** You don't ever want your nonces to be poly.

**Leo:** I know what a nonce is, and I don't want to be messed with. No way.

**Steve:** You don't want polynonces.

**Leo:** Unh-unh. Is nonces.

**Steve:** This is not the Polynesian attack. This is the polynonce attack.

**Leo:** Okay.

**Steve:** Just to be clear. The guys at Kudelski Security Research, we've spoken about them before, they've discovered a new and novel attack which they call "polynonce," and we'll see why here shortly, which enables this attack. Okay. It is available in instances where a weak pseudorandom number generator was used to create a, well, was used in the use of a digital signing algorithm. This allowed them, among other things, to recover the private keys for hundreds of bitcoin wallets which were for transactions that were signed using weak pseudorandom number generators.

Okay. So I'm going to sort of give away the story here in their upfront summary, but I think that's a good thing to do because then you will understand where we're headed with this because this is going to be a bit of a deep dive. You know, make sure that your propeller spring is wound up on your beanie cap.

So they said: "In this blog post we tell a tale of how we discovered a novel attack against ECDSA" - I'll explain what that is in a second and why it's important because it's everywhere - "and how we applied it to datasets we found in the wild, including the Bitcoin and Ethereum networks. Although we didn't recover Satoshi's private key," they said, "we'd be throwing a party instead of writing this blog post."

**Leo:** Yeah. They'd be on an island somewhere.

**Steve:** That's right. They said: "We could see evidence that someone had previously attacked vulnerable wallets with a different exploit and drained them. We cover our journey, findings, and the rabbit holes we explored. We also provide an academic paper with the details of the attack and open-source code implementing it, so people building software and products using ECDSA can ensure they do not have this vulnerability in their systems." And how.

They said: "Part of the Kudelski Security Research Team's activities includes looking into new vulnerabilities and exploits. A few months ago, while researching ECDSA nonce attacks, a member of our team discovered a more general way to exploit complex relations between nonces to retrieve the signing key." And I'll explain a lot of this here as we go along. They said: "A review of existing literature seemed to confirm that this was indeed a novel insight, so we started digging into it. If you're interested in the math and details surrounding the attack, there's a link to the paper."

Okay. So we're about to be hearing a lot about ECDSA. ECDSA is an abbreviation for Elliptic Curve Digital Signature Algorithm. It is now, even still, the current state-of-the-art choice for efficiently producing a secure signature of a hash of a message where the signer's key is kept secret, and they publish their matching public key, which is then used to verify their signature. So ECDSA is used everywhere. It's the algorithm used to sign Bitcoin, Ethereum, and other cryptocurrency blockchain transactions. It's one of the signature algorithms available to TLS connection handshakes when a site is using an elliptic curve certificate. So the exploitation of poorly chosen random nonces could and would have far-reaching consequences.

Okay, so I'm going to read what they wrote, and not all of it because they really get into the weeds later. And not because I expect anyone to exclaim "Oh, yes, of course, how obvious!" No. But because it will give you some sense that this was not trivial, and that these guys are geniuses. Oh, and if you're listening to this while operating any heavy equipment, please consider pausing your work during this rough patch.

Okay. So they wrote: "In a nutshell" - and, yeah, this is the nutshell version. They said: "The attack looks at the fact that you can always define a recurrence relation among nonces used in different ECDSA signatures as a polynomial of arbitrarily high degree, with unknown coefficients, modulo the order of the curve's generator point." We're going to take their word for that. They said: "If you have a set of N ECDSA signatures under the same private key, and this recurrence relation is of degree D, then under some caveats we'll talk about later you can use the ECDSA signature equation to rewrite the polynomial in terms of the private key and the recurrence unknown coefficients." Okay, that was the insight that one of these guys had. Obviously they're way deep in the weeds of this.

So they said: "We have found that the unknown coefficients can be eliminated from the polynomial, which always has the signer's private key among its roots. So, if the degree D is low, and you have enough such correlated signatures where the number of signatures is N, and you need N to be equal to D+3 at least, then you can perform a key recovery attack" - and of course that's the key, the key term here. This all boils down to a key recovery attack. They say: "...by simply finding roots of a polynomial with known coefficients over a finite field," which they note is an easy task on a computer.

"To run the attack in practice," they said, "the following is required: a minimum of four signatures generated by the same private key, the associated public key, the message hash associated with each signature." And note that all of those will be available anywhere that ECDSA is used. And they said: "If the nonces" - what are supposed to be pseudorandom, completely random. They said: "If the nonces obey the recurrence relation, we retrieve the private key used to generate the vulnerable signatures."

They said: "The more signatures used in the attack, the slower it gets, but the more likely it is to succeed." Because, you know, they've got more information. They said: "If you attack N signatures with their N nonces, follow a recurrence relation of degree at most N-3." And the higher the degree of the polynomial, the slower and more difficult the attack. But they said: "If you attack N signatures with their N nonces with a recurrence relation of degree at least N-3, then you can perform a key recovery attack on ECDSA!" they wrote, because that's their discovery.

Okay. So it's big news because it means that the choice of repeated nonces, that is, the choice of the nonce which is used as part of elliptic curve digital signing, must be of high quality. For any given repeated signer their chosen nonces must really be random. And, by the way, must absolutely never be reused. Reusing a nonce is like the biggest no-no.

**Leo:** Is this related to - remember when NIST kind of cracked an earlier elliptic curve function? Not cracked it, but provided constants.

**Steve:** Yeah.

**Leo:** Right?

**Steve:** It was the DRB, the deterministic random bit generator, that BSAFE had that RSA was publishing as their default PRNG, and it was shown not to be that good.

**Leo:** Because they used starting seeds that the NSA had specifically chosen.

**Steve:** Yeah, they used magic numbers and didn't tell anybody where they came from.

**Leo:** So this is not the same thing at all.

**Steve:** Not the same thing.

**Leo:** Okay.

**Steve:** But so far what we've learned is that the ECDSA is everywhere. It is the preferred fast, small, digital signature algorithm. And where it is known to be exquisitely sensitive to the quality of the nonces, a nonce must be chosen for each signature, and they have to be chosen well. What these guys have - their inspiration was they basically came up with the attack of, if the nonces were not being chosen well, now we know actually how to take advantage of that.

**Leo:** Ah. I wonder if I can get in my wallet this way. This is interesting. Okay.

**Steve:** So you can sort of think of it as the price of doing business with ECDSA. Yes, it's a terrific algorithm. It's super secure using very short keys. But it's also quite finicky. It's well known that even a single inadvertent reuse of an ECDSA nonce will completely expose the signer's private key. You've still got to do a bunch of math, but it is available. So you reuse a nonce, and you sacrifice all of the money in your cryptocurrency wallet.

**Leo:** Oh, this wouldn't be useful for me, though, because they can't choose the wallet that they want to look at. They need to look at all of them and look for matches; right? Is that kind of the way they - you couldn't just give them the wallet and say is this easy to crack?

**Steve:** If you ever transacted with that wallet, if you...

**Leo:** I have, yes.

**Steve:** Then there's a possibility because the earlier uses, which is when you and I both were messing with this in the beginning...

**Leo:** Mm-hmm, long time ago. That's why I forgot the password, yeah.

**Steve:** There tended to be a reuse of private keys. In theory, ECDSA you should have an ephemeral private key also for each signature. You should just make one up because you're able to then tell people what the public key is. You keep the private key secret. But you don't need to always use the same private key. Back in the beginning we were tending to do that. You were probably doing that.

**Leo:** Yeah. I was using the bitcoin default.

**Steve:** Bitcoin Core, I think.

**Leo:** Bitcoin Core, yeah.

**Steve:** Yeah.

**Leo:** Oh, good. There's hope.

**Steve:** They said - yeah.

**Leo:** This may be a quarter of a million dollar show for me. Okay, keep going, keep going.

**Steve:** They said: "We tested the attack on a specially crafted set of signatures to verify that it works. It does. In simpler words," they said, "what our attack means is that every time an ECDSA signature is generated, the signature itself gives us a relation between the nonce and the private key. If the nonces are truly randomly generated, this should never be a problem because the chance that a number of nonces picked at random fit on a low-degree polynomial recurrence relation is negligibly small.

"But there's a catch," they said. "Nonces are usually output by a pseudorandom number generator rather than being really random, and PRNGs are deterministic algorithms with relatively low complexity. In the best scenario, the PRNG used is complex enough and cryptographically secure, meaning among other things that any polynomial correlation between its outputs will have such an astronomically large degree that you can safely consider it indistinguishable from truly random.

"But," they said, "weak PRNGs are basically everywhere. Take, for example, the simple case of a linear congruential generator (LCG), which is the typical textbook introduction to PRNG implementations. LCGs are to PRNGs what ROT13 is to encryption and '123' is to secure passwords."

**Leo:** Okay. Good, good.

**Steve:** In other words...

**Leo:** Not good.

**Steve:** Not good. And they said: "Despite that, due to their simplicity and popularity, they're the default choice for many non-critically secure applications." And even that I would be skeptical of. And they said: "And it is totally possible that a 'placeholder' LCG implementation slipped into production code without being replaced by a secure one."

Okay, now, I'm going to interrupt. I'm going to do a little segue here for a minute because I happen to like linear congruential number generators. So I'm going to talk a little bit about them and how I just used one. And then we'll get back to this. So they are very handy for many purposes. But in no way imaginable would anyone consider them secure for any purpose. The entire LCNG (Linear Congruential Number Generator) algorithm is to simply take the current seed, the previous number generated; multiply that by a fixed number whose binary representation has a complex bit pattern, and sometimes a prime number is used, although it's not necessary; then add a second constant value. That's it. And that's its attraction. A single multiplication followed by a constant number, followed by a single addition by a different constant...

**Leo:** It's not random at all.

**Steve:** Oh, god, no, Leo.

**Leo:** It's just not obvious where it came from.

**Steve:** Exactly. Basically it's jumping forward in its number space by a certain amount. So, you know, on the negative side it's entirely predictable. Given any output, it's possible to immediately produce all future and even all previous outputs, if you run it backwards. You know? It's awful. So as a source of randomness, the term "random" doesn't apply at all. There should never be an "R" anywhere near it. What it has going for it, however, is speed, since nothing could produce a noisy pattern of data faster. Which is why, as it happens, SpinRite received one a few months ago. And since I think everyone will find this interesting, I'll take a minute to explain.

A couple months ago we hit an unexpected problem with SpinRite, when some very troubled hard drives would encounter some damage they could not handle. Now, the proper behavior would be for the drive to stop the transfer and return an error code along with the number of sectors that were successfully transferred prior to that error, prior to encountering that error. That's in the API. That's in the spec. That's what hard drives are supposed to do. And that would allow SpinRite to then zero-in on the one sector that tripped it up, and then it would work to recover that sector's data. That's what SpinRite does.

So as I said, that's what drives are supposed to do. But it turns out that many drives which our testers have will just hang and wait for SpinRite to give up and timeout the transfer. Or sometimes they'll return a "command aborted" error without any indication of where they were when they became upset. So now we had a problem. Previous versions of SpinRite, all previous versions of SpinRite, which did all of their work through the system's BIOS, were therefore limited to the BIOS's maximum 127-sector transfers, which is about 64K. So SpinRite would drop out of transferring those 64K blocks at a time and switch over to single-sector mode to locate wherever the trouble that might be occurring at that particular sector. So it would just switch over and do them one sector at a time to find the problem.

Well, that worked great for finding one or more troubled sectors within a 127 possible problem sector block. But 6.1 achieves its tremendous speed increase by transferring

32,768 sectors at once, which is 16MB rather than 64K. So dropping out of that to switch over to one sector at a time was really no longer an option. I mean, it would take forever. So the solution to this dilemma is why I added a simple linear congruential number generator to SpinRite.

SpinRite zooms along, running at the maximum speed that the drive can go, transferring 16MB at a time until it hits a problem. If the drive returns the number of sectors that were successfully transferred, as it should, then SpinRite immediately zeroes in on that sector, works on its recovery, reaches a conclusion, then resumes the interrupted transfer with the next sector after the one that caused the interrupt. But if the large transfer stumbles without indicating where among those 32,768 sectors that were requested, SpinRite uses its new linear congruential number generator.

What it does is it fills that 16MB transfer buffer with one sector of noise pattern repeated over and over, 32,768 times. Now SpinRite has a known pattern of noise throughout the transfer buffer. So it re-requests the same 16MB transfer. And this time, when the drive craps out without saying where, SpinRite can determine that for itself. It scans forward through that transfer buffer searching for the first sector that matches the noise sector pattern. That will be the first sector in the transfer that was not overwritten by the drive's successful reading of sectors, so that's where the transfer was interrupted. And that's the sector that SpinRite then needs to work on recovering.

**Leo:** That's very clever. That's cool.

**Steve:** It's so cool.

**Leo:** But it has to be random or you wouldn't be sure that you were on the right place.

**Steve:** Exactly. It has to be random or you wouldn't be sure that the drive might not have contained that data, that sector of data. And you'd get a false positive match. So if anyone's been wondering how all this time is being spent, there's an example. 6.1 is going to be far better than any SpinRite that ever existed before, which is why I'm getting so excited that it's also getting very close.

So that's a perfect example of where a linear congruential number generator can be useful and come in handy. Because, for example, there we have zero need for security. You just don't even consider it a random number generator. We just want noise. Okay. So picking up on this team's investigation, they said: "The bitcoin blockchain" - oh, I love this, Leo. "The bitcoin blockchain is basically a large public mine of ECDSA signatures."

**Leo:** Oh.

**Steve:** "In fact," they wrote, "ECDSA has been used as the default signature algorithm since Bitcoin's creation in 2009. We know that, in principle, most ECDSA signatures in the Bitcoin network are ephemeral" - that's what I was mentioning before - "in the sense that the generating secret key is only used once, but we also know that this practice is not always in place, especially for the older transactions. And also Bitcoin has the advantage that the blocks follow a temporal history, which puts a certain degree of order on the signature generation, although that's only approximate because there's no way to

determine the order in which signatures in the same block were generated since the timestamp is only recorded for a block, not per signature."

They said: "The problem is that these are mainly speculations, and we have no clue how accurate all these speculations are. So it's time to verify. We downloaded," they wrote, "and installed Bitcoin Core, the official Bitcoin client, and let it synchronize the whole chain." Get a load of this. "The sync process took about a day on a fast fiber connection, and the total blockchain size was about 430GB, up to block number 752,759 on September 5th, 2022."

**Leo:** It's bigger today, yup.

**Steve:** Uh-huh. "Dumping all the signatures and original messages from the raw blockchain data took 24 hours. The resulting output file size was 271GB and contained 763,020,390 unique signatures. This file contained, on each line, the output address, the Elliptic Curve DSA signature R and S values, public key, transaction ID, original message, and block timestamp. We grouped the signatures by public key and then, within each group, sorted signatures by timestamp to have more chances of picking consecutive ones. At this point, we had a dataset ready to run the attack. But first, here are some statistics about the dataset." And actually these are important, so I included them. They said: "These signatures were produced by private keys associated with 424,549,744 unique public keys." Okay. So 424 million-plus unique public keys. Of those 424 million public keys, 390 million, or about 92%, produced only one signature.

**Leo:** That's good; right?

**Steve:** Yes.

**Leo:** Okay.

**Steve:** Exactly. Remember that for Bitcoin, private keys are supposed to be ephemeral and are supposed to only be used once. So it's to be expected that 92% of all those signatures would use a completely unique key.

**Leo:** Well, you'd like it to be 100%.

**Steve:** You would. That means there's still hope here.

**Leo:** Okay.

**Steve:** And so that means that there's no attack possible on those since these guys are looking for weak nonces occurring during key reuse with the same private key. So they needed to find instances where multiple signatures used the same key. And they did. They wrote: "There were 34 million public keys with at least two signatures, used in at least two signings, two signatures; 18 million with at least three signatures; 12 million with at least four signatures; 9.6 million with at least five signatures, and 7.8 million with at least six signatures."

**Leo:** That means six seen elsewhere, five seen elsewhere, duplicating six other signatures.

**Steve:** Correct.

**Leo:** Yeah, okay.

**Steve:** Yes, yes. So that is to say the same private key was used to sign six transactions.

**Leo:** You have a collision with six transactions when you'd expect to have no collisions in trillions of transactions; right?

**Steve:** Right, right. So they said there was a considerable number of public keys with over 200,000 signatures.

**Leo:** Ooh. Ow.

**Steve:** Yeah. Somebody just didn't ever change their public key, or their private key, because the private key and the public keys match; right? It's a one-for-one relation. The public key associated with the most signatures had 3.4 million signatures.

**Leo:** Oh, my god.

**Steve:** Yeah.

**Leo:** So there was one key shared by 3.4 million accounts. They had the same public key.

**Steve:** One key used in 3.4 million transactions.

**Leo:** Transactions, not accounts, okay.

**Steve:** Right, transactions. So they said the attack is generic and can be run with at least N=4, that is to say, where you've got a single private key used in at least four transactions, N=4 signatures, and they call that the "linear case." That is to say, if N=4, then they're solving a linear function.

**Leo:** Okay.

**Steve:** But can also be run with more signatures. And the more you have, the better your chances.

**Leo:** Of course, yeah.

**Steve:** They said, for example, five signatures for the quadratic case and six signatures for the cubic case, or even more signatures.

**Leo:** So it gets easier to crack with more signatures.

**Steve:** Actually it gets harder to crack but more likely.

**Leo:** Ah, okay.

**Steve:** Yeah. So they said: "The linear case will also detect repeated nonces, but it is more general because it can exploit any linear recurrence relation." They said: "However, we wanted to go even further and run the quadratic case with N=5 because we thought it might give more interesting results. We considered the cost/benefit ratio of performing cubic or higher order attacks to not be worthwhile, so we stopped at the quadratic case."

**Leo:** So they cost too much compute time.

**Steve:** Right, exactly, meaning batches of five signatures. But notice if you're doing five at a time, but if you had seven signatures, you could slide a window along those seven, taking them five at a time; right? So they said: "Since we sometimes have more than five signatures associated with a given public key, we decided to perform a sliding window over the signatures sorted by timestamp and run the attack on each window of size N, where N=5. So," they said, "how did it go? We ran the sliding window attack with N=5 on a 128-core VM, and it was completed in two days and 19 hours. The estimated cost of the attack was about 285 USD."

**Leo:** Probably did it in the cloud, okay.

**Steve:** They did. "We broke 762 unique wallets."

**Leo:** Was mine in there? Please?

**Steve:** I don't think so because, although you probably haven't checked yours...

**Leo:** No. I don't know how to run this attack.

**Steve:** Right. "All of these had a zero balance."

**Leo:** Oh. Oh.

**Steve:** Because they were not the first people there, Leo.

**Leo:** Oh. They'd already been drained.

**Steve:** Yes. They said: "Interestingly enough, we could break all these wallets, not because of a linear or quadratic recurrence, but because there was at least one repeated nonce in the signatures. So," they wrote, "it looks like the common mishap of ECDSA implementations using a repeated nonce was the cause of the trouble."

**Leo:** So they went to all this effort to try to fight a very sophisticated attack.

**Steve:** Yeah.

**Leo:** And what they really uncovered was there was a trivial attack which had already been uncovered and used.

**Steve:** Exactly that.

**Leo:** Wow.

**Steve:** In other words, these guys developed a highly sophisticated and subtle attack, which they showed would have worked, and which would have been able to detect subtle failures in nonce choice. But what they stumbled upon during this work was the biggest no-no in the use of Elliptic Curve DSA, which is a single reuse ever of a nonce when signing under the same private key.

**Leo:** Wow.

**Steve:** This meant that a trivial attack against those wallets was possible; and what's more, somebody had already done that. And Leo, they tracked them down.

**Leo:** Oh, my. Oh, my.

**Steve:** Because the bit, you know, it's a ledger; right?

**Leo:** Because you transfer the stuff from this account into another account.

**Steve:** Exactly.

**Leo:** And that account number is not private. It's not hidden. It's public.

**Steve:** Exactly. It's a public ledger.

**Leo:** This is exciting. This is great. We could have made a whole podcast series out of this. This should have been the Security Now! Season 20. This could have been amazing. Keep going. This is great.

**Steve:** They said: "Since we only ran the attack using a window of size five so far, we may have missed a few vulnerable wallets that would only have been found for public keys that had exactly four signatures. So we reran the attack with N=4 on only the signatures from wallets with exactly four signatures. We were able to break 11 new wallets with a zero balance and at least one repeated nonce, thus increasing the total amount of broken wallets to 773." So in other words they didn't find any new wallets because, again, they were finding a superset, and it turned out, well, there were other repeated nonce instances. So unfortunately, there were a bunch of people out there who had a bad, you know, who had a bad bitcoin client that was reusing nonces when they were doing transactions. And they lost all their money.

They said: "We suspect, and in some cases have evidence," and they said, "as we will discuss later, that all these wallets have zero balance because they have already been hacked in the past due to the repeated nonce vulnerability. We also estimated the total theoretical amount of tokens that may have been stolen from these 773 wallets to be 484 BTC, at a value of approximately 31 million USD at Bitcoin's peak."

**Leo:** Well, one thing that's good is, since I know there's still bitcoin in my wallet, I'm not one of those...

**Steve:** You're not, yes.

**Leo:** ...duplicated nonce wallets.

**Steve:** Yes.

**Leo:** Yeah, okay.

**Steve:** So anyway, then they go on at length talking about various other attacks. Those are the rabbit holes that they refer to. And they talk about all the evidence of, you know, they identified a number of actors, and they found the person who, you know, the bitcoin address that essentially made these transfers. Anyway, they conclude, saying: "So, since we aren't sipping Mojitos on a beach in some exotic location, you can tell we did not gain access to Satoshi's wallet. But we recovered..."

**Leo:** Well, they might have, it's just that somebody got there first. It could - I guess not.

**Steve:** Yeah, actually, doesn't Satoshi have an insane number of bitcoins?

**Leo:** Oh, yeah, billions, billions and billions of dollars, yeah.

**Steve:** So they said: "We recovered the private key of some Bitcoin wallets showing that the attack works. We only scratched the surface by looking at Bitcoin, Ethereum, and some TLS connections. With this initial look, we wanted to ensure that an attacker could not cause financial damage before releasing the details. But there are many other locations where Elliptic Curve DSA is used, such as additional blockchains, batches of PGP signatures, other TLS connections, and embedded devices, just to name a few. We release this information along with code so that people and organizations can proactively ensure they are secure against these attacks and create more robust systems. We hope you find this useful."

**Leo:** What would be your fix for this? Somebody's asking in the chat room. It wouldn't be changing your password. You'd probably transfer it out of your old wallet, make a new wallet, and transfer it into a newer wallet where presumably this nonce won't be reused.

**Steve:** So, yes, so you absolutely, well, so nonce reuse is never a problem as long as you're not reusing your private key. The easy thing to do is make sure you get a new private key.

**Leo:** Oh, rekey it.

**Steve:** For every transaction.

**Leo:** Okay. Okay.

**Steve:** Yes. And that's why 93%, or 92% whatever it was, of all transactions are using unique private keys. Everyone should have a client doing that.

**Leo:** Okay.

**Steve:** In the earliest days, people were not doing that.

**Leo:** So these are all older wallets.

**Steve:** Yes. Well, older transactions.

**Leo:** Transactions; right.

**Steve:** And so those would have been older transactions using older bitcoin clients.

**Leo:** I wonder if Bitcoin Core now, I'm sure it now rekeys each time, I would think.

**Steve:** Given that 92% of all transactions are, that must be the case. And they're supposed to be. But they're, you know, they're not always.

**Leo:** Right.

**Steve:** You know, again, everybody, there's like all these different bitcoin gizmos out there. And some of them just aren't well made. And some of them apparently, not only are they not bothering to use a new private key for every transaction, they're also, who knows, how can you reuse a nonce? How could you get anything that you thought was even - that you even thought was supposed to be random and get it twice. Wow.

**Leo:** Wow, wow.

**Steve:** So, you know, there are a couple of takeaway lessons here. One is that details really matter; right? The crypto gurus who invent and create these algorithms admonish their implementers that, if they want to take advantage of the elegant, though quite finicky, ECDSA algorithm, they're going to really need to be double-damn certain to only ever use it with truly high-quality random nonces in an application where they need to have a static signature. And most Elliptic Curve DSA signatures are static; right? If you have an Elliptic Curve TLS certificate, that's not changing. So you're depending in that instance on the quality of the nonce being highly random. But so the point is, yes, it's a great algorithm, but you've got to use it correctly.

And the second takeaway, I think, is in the form of a question. If details like that really matter that much, if a critical algorithm is so brittle and sensitive to difficult-to-control implementation mistakes, should it be chosen and used in critical applications? You know? Because it's no longer the case that only one of these is ever made. Bitcoin went crazy, and everybody said, oh, I'm going to create a wallet. I'm going to create my own because why not? Let's write it in JavaScript. So should it be chosen and used in critical applications such as to protect the storage of hundreds of billions of dollars in cryptocurrency wealth? And by the way, I was curious. If anybody else is, as of the beginning of this year, total wealth stored in cryptocurrency was $804 billion, with around 320 billion of that in Bitcoin.

So this is the sort of way these things happen; right? We see it over and over. The inventors of the packet-switching Internet could have never foreseen what their experiment grew into. And, yes, for all that it's amazing, it also has some weaknesses in the face of deliberate abuse. And with Bitcoin, as a proof of concept, Satoshi invents and designs an intriguing system. And he chose Elliptic Curve DSA to sign transactions because, sure, it's the best, though it also needs to be handled with extreme care. Little could he have possibly known what his experiment would grow into. And in retrospect, choosing something less fragile, some less fragile crypto would have probably been a better choice for something that was to grow into a global phenomenon. And of course he could have never predicted what was to come.

**Leo:** So I use Curve 25519. That's an elliptic curve, but that's not - it's different.

**Steve:** That's probably not using DSA.

**Leo:** Right.

**Steve:** Well, although it is the default signing. So we'd have to look and see which of - so 25519 is one particular very popular elliptic curve. There are other elliptic curves. But all of them then share the same set of algorithms which are put on these different elliptic curves.

**Leo:** So I use it with SSH primarily. I shouldn't be announcing this, but that's what I primarily use it for.

**Steve:** Yeah, I mean, that's what SQRL uses.

**Leo:** Yeah.

**Steve:** I also chose 25519.

**Leo:** It's also because the keys are short and easily copied and pasted.

**Steve:** In my case it was because it had some particular properties that Dan Bernstein, who is the inventor of 25519, it's got some really cool properties that enable the use of, you know, enable SQRL's crypto.

**Leo:** Yeah. Okay. I won't worry.

**Steve:** Very cool.

**Leo:** And since my coins are still in my wallet, I guess I don't have to worry about this attack, either.

**Steve:** No.

**Leo:** Well, doesn't really matter, I can't get to them anyway.

**Steve:** I think we're at an hour and 18. Let's take our third break.

**Leo:** Yeah, let's take a break. Yeah, yeah, yeah.

**Steve:** And then we will continue.

**Leo:** Yeah. And, you know what, if the wallet had been emptied, I would say, well, I'm glad somebody got the use of it. Oh, lord. What a world, huh?

**Steve:** Now, there is no relation between PlexTrac and the Plex Media Server.

**Leo:** No. Oh, let's hope not.

**Steve:** Just want to make that clear.

**Leo:** Oh, no.

**Steve:** Last week we were talking about the growth of CISA's KEV database, where KEV is the abbreviation for "Known Exploited Vulnerabilities"; and how, while it grew much faster last year than in any previous year, an examination of the dates of the CVEs that were added during this most recent past year revealed that the large majority of these were not new problems being exploited, but rather old problems that had never been patched. So I noted that CISA had just added CVE-2020-5741.

**Leo:** 2020. That's three years old.

**Steve:** That's right. And what is 5741, you might ask? Well, it's a deserialization flaw of untrusted data which was found, as we noted, three years ago in the Plex Media Server for Windows.

**Leo:** But who would be - who would be running three-year-old unpatched versions of Plex on their Windows machines? Who indeed?

**Steve:** Who indeed, my friend. It happened that an unfortunate LastPass developer...

**Leo:** Oh.

**Steve:** Yes, was doing so, after which a distinct lack of fortune was visited upon all LastPass users.

**Leo:** That's a good way to put it.

**Steve:** A remote, authenticated attacker is able to execute their arbitrary Python code on the victim's computer.

**Leo:** Can we make it any easier? Holy cow.

**Steve:** So now we know that this developer was using a publicly exposed Plex Media Server which was three years out of date, since of course CVE-2020-5741 had been found, was known, and had been fixed.

I've been saying for a while now that any serious cyberwarfare agency or group across the globe must be maintaining a vulnerability and exploit database indexed by target vendor. So in the instance of this second LastPass attack, LastPass's developers were identified, probably with the aid of the first attack on that developer network; right? Then they were tracked down and identified at home, and their home IP addresses were scanned. When port 32400 was found to be accepting inbound TCP connections at one of those IPs, that port was looked up, and the Plex Media Server was found to be the most common user of that port. Then the attacker's master vulnerability and exploit database was queried for "Plex," and a three-year-old remotely exploitable vulnerability stood out. Could we be this lucky, the attackers probably thought to themselves. And indeed, they were, and we weren't.

**Leo:** I know a little Python. I think I could take advantage of this, guv'nor.

**Steve:** That's right. And I don't know if anything more has been learned. But I did hear something about North Korea...

**Leo:** Oh, really.

**Steve:** ...being the presumed source of the attack, yeah.

**Leo:** The thing that makes me scared, it was so clearly targeted against LastPass.

**Steve:** Oh, boy, yeah.

**Leo:** And that means they knew what they wanted, which is the vaults. Which means, I presume, they knew what to do with them.

**Steve:** Yeah, yeah, which is the reason we're no longer all using LastPass.

**Leo:** Yeah.

**Steve:** Okay. One quick random note, and then we're going to talk about Sony suing Quad9. I wanted to mention "Andor," Leo.

**Leo:** Yeah. It's good; isn't it?

**Steve:** In a word, "Wow."

**Leo:** Yeah.

**Steve:** And of course there'll be no spoilers here. But I want to set the stage a little bit and give our listeners just a bit of background to raise their curiosity level. And if you're

someone who wants to hear absolutely nothing in advance, then skip forward 60 seconds.

"Andor" presents the story of an orphan, Cassian Andor. The series is set well before the events of Luke and his princess sister. It tells the story of the early rise of the Empire as it gradually displaces corporate rule for imperial rule and tightens its grip on the galaxy's citizenry, which is increasingly becoming stratified into upper and lower castes.

Mostly, Cassian just wants to be left alone. He grew up hard and survives by reselling tech that he steals from around the fringes of the Imperium. Despite being raised by an adopted mother who has been seeing what's happening to the galaxy and wants to rebel, he has zero interest in any "cause." He holds no such ideals. He just wants to be left alone and not be told what to do. Over the course of these first 12 beautifully crafted episodes, we watch that change.

If you don't already subscribe to Disney+, I cannot imagine that you would regret subscribing just long enough to watch this first 12-episode season. There might even be a free trial period available to new subscribers. I was not offered one because a year or two ago, after watching the first season of "The Mandalorian," I canceled my subscription. The Mandalorian wasn't horrible, and Lorrie loved Baby Yoda. I kept hearing, "Ohhhhhhh." But neither did it strike me as being all that great.

**Leo:** It's just a rubber puppet.

**Steve:** "That's so cute, ohhhhhhh." Anyway, but "Andor" is another thing entirely. And Leo, I think that this very sober and serious series may be the best Star Wars property I've ever watched.

**Leo:** I agree. And I think it is because it's very little Star Wars, in a way; right?

**Steve:** Right. There's no - who was that oosa woosa goosa thing?

**Leo:** No, no, none of that stuff.

**Steve:** The floppy ears.

**Leo:** No Jar Jar, no.

**Steve:** There's no Jar Jar Binks. There's no Ewoks.

**Leo:** I haven't seen the last episode yet, but I don't even think there's any light sabers. Maybe there is towards the end, but...

**Steve:** Oh, but Leo, that fight scene in the second-to-last episode?

**Leo:** Oh, so good.

**Steve:** Oh, my god.

**Leo:** So good, yeah. And so it's - a number of people have told me this. It's a good Star Wars show if you don't really want a Star Wars movie. But there is that undercurrent which you mention of we're seeing the rise of the Empire.

**Steve:** Yes.

**Leo:** So if you're a Star Wars fan you'll like it. And there's a lot of, you know, there's TIE fighters. There's Star Wars technology in it. But it's a good story.

**Steve:** And as Lorrie said, it has a lot of pew pew pew, pew pew pew pew, pew.

**Leo:** It does. And the storm troopers are still lousy shots. But okay. You know, oh, yeah, I thought it was really, really, really good, yeah. Very, very good.

**Steve:** Anyway, so we talked about "The Expanse." We loved "The Expanse." And again, if you hate Star Wars, I would say, I think you're right, Leo, forgive it for being set in that universe.

**Leo:** Yeah. Because it's barely Star Wars, yes.

**Steve:** And the other thing I thought was really interesting is the rebels are shown to be - they're just as ruthless as the Empire in their own way. I mean, it's not, you know, sugarcoated and lollipops.

**Leo:** It's more of a gritty kind of real...

**Steve:** Yeah.

**Leo:** Somebody says it's like a spy story. It is. It's a little more like it would be - if you took away the Star Wars trappings, it'd be the same story, same show.

**Steve:** Oh, and boy, there are some people you love to hate.

**Leo:** Oh, yeah.

**Steve:** I mean, they're just beautiful characters.

**Leo:** Oh, there's some bad people in there.

**Steve:** Yeah. Yeah. So by all means, watch the last one. It has a great ending. You won't be disappointed.

**Leo:** Yeah, yeah. No, I've been saving it.

**Steve:** And I just can't wait for the next season. It's like, oh, come on, let's keep going. Yeah.

Okay. So I chose the news of Sony's lawsuit against a well-known and well-respected DNS provider because a very dangerous legal precedent threatens to take hold on the Internet. Here's how the defendants, Quad9, summarize the situation and its danger. They wrote: "Sony Music Entertainment Germany is litigating against Quad9, requiring us to block access to a website that links to a site containing files that Sony asserts are violating their copyright. We maintain that Sony's request essentially amounts to content censorship and risks cracking the foundations of a free and open Internet, in Europe and potentially worldwide. Censorship, in turn, can lead to undue restrictions on freedom of speech."

I'll just briefly say that for those of our listeners who were not here, Leo, for Episode 12 of this podcast, this is not the first time Sony has earned some negative coverage and been named in this podcast's title.

**Leo:** I'm starting to hate them, to be honest.

**Steve:** Episode 12, dated November 3rd, 2005 ran all of 24 minutes. Its title was "Sony's 'Rootkit Technology' DRM (Copy Protection Gone Bad)."

**Leo:** Twenty-four minutes, really?

**Steve:** You said, hey, Steve, why don't you come, you know, maybe just like half an hour, and tell us what's been going on every week.

**Leo:** Not much to say, but we do it in half an hour, yeah.

**Steve:** Since those early days, Sony has had their troubles. And they do tend to be litigious when they can find someone, anyone, to sue. They're happy to throw their weight around without much provocation. So now to this issue today.

We've talked about Quad9 in the past. They're a bunch of good people. Quad9 is a free global public recursive DNS resolver. Meaning that anyone in the world can ask them to look up the IP address associated with a domain, and they will do that by asking whatever other DNS resolvers may be needed to come up with a final answer. Thus recursive. But Quad9 is also explicitly a filtering DNS resolver. In fact, that's its purpose. It aims to protect its users by not returning the IP addresses of sites known to be malicious. Quad9 is operated by the Quad9 Foundation, a Swiss public benefit, nonprofit foundation headquartered in Zurich, Switzerland, whose sole purpose is improving the privacy and cybersecurity of Internet users. Quad9 has 200 points of presence globally spread around 90 countries from which they provide these services to individuals and organizations at no charge.

Here's the language from the court which supports the position that Sony's attorneys have described. This is taken directly from the official court order. They said: "The defendant is ordered to refrain from selling on the territory of the Federal Republic of Germany the music album 'Evanescence - The Bitter Truth.'"

**Leo:** Fine. Fine.

**Steve:** Exactly. Okay. Don't want it.

**Leo:** Don't want it.

**Steve:** Happy not to sell that, "with the sound recordings contained thereon, to be made publicly available, by the defendant providing its users with a DNS resolver service, which the domain 'canna.to' and/or the subdomain 'uu.canna.to' it translates into numeric IP addresses, so that it is possible for the users of the defendant, with the help of these numerical IP addresses, to reach the Internet service under the domain 'canna.to' and/or the subdomain 'uu.canna.to' and/or the further domains and to call up their links to unlawful storage of the album, as happened by the defendant offering its users the DNS resolver service Quad9 at the IP address 9.9.9.9, with the help of which the users with the links" - and then the court order has four links. Two of them are uu.canna.to links, you know, which some specification in the URL, and canna.sx and canna-power.to.

They said: "Numeric IP addresses were transmitted which enabled them to access the hyperlinks provided at the aforementioned addresses to the storage locations of" - and then there's two links at shareplace.org and some hex stuff. So that's the actual pirate site, shareplace.org, that is offering up this material. And they finish: "And call up the illegally stored copies of the aforementioned album." So in other words, there's no misunderstanding here with the court about the essentially passive role that DNS and a DNS service provides. Another piece of the official proceedings was also interesting. They described the efforts that Sony first went through in doing what we would all agree was the right thing for them to do. Here's what the court document describes.

They said: "Music content is listed and categorized under the domain www.canna.to. The domain of the website is 'CannaPower.' In a letter dated March 23rd, 2021" - okay, so note that this has been going on for two years - "the plaintiff," meaning Sony, "drew the defendant's," meaning Quad9, "attention to the infringement and also pointed out the URL. The defendant was requested to put an end to the infringement. The defendant [Quad9] was warned after it failed to remedy the situation. The plaintiff [Sony] had made every" - here it is. "The plaintiff had made every conceivable effort to remove the infringing offer with the involvement of primary liable parties. The CannaPower website has no imprint. Entries on the domain owner were also not available." Oh.

"Requests for deletion to the hosting provider went unanswered." Oh, no. Nobody answered the phone. "Their two IP addresses were named. The company Infium, UAB with an administrative and technical contact in the Ukraine was identified as the responsible organization. The company was allegedly based in Vilnius, Lithuania. There, however, a delivery by courier could not take place due to the lack of a traceable signature. In Ukraine, delivery was not possible because the address was located in a high-security area to which it was not possible to accept deliveries without express consent; this consent had been refused."

Okay. So in other words, the actual copyright infringers are out of reach of Sony's wrath. From what the court wrote, it doesn't really sound as though Sony tried that very hard to

go after the primary sources. The court wrote that Sony had made "every conceivable effort to remove the infringing offer with the involvement of primarily liable parties." But CannaPower's hosting provider, what? Didn't answer the phone? So that was it.

**Leo:** No one's there. We can't serve them.

**Steve:** So Sony, unwilling to be stymied and denied, decided to attack someone who was within their reach. The court wrote: "The plaintiff is of the opinion" - that is, Sony is of the opinion" - that the defendant is liable as a tortfeasor." I had to look that one up. The short version of the definition of tortfeasor is: "A tortfeasor is a person or entity who is found to be responsible under civil law for an injury caused to another person or entity."

In other words, Sony is claiming that because Quad9 provides some of the Internet glue mechanics that are required for users to reach the CannaPower website given its domain name - and note, only Quad9 users, not all the rest of us - they, Quad9, are thereby responsible under civil law for the injury caused to Sony. Wow. The court wrote: "According to recent case law, it" - meaning Quad9 - "is also liable as the perpetrator of a copyright infringement."

Okay, now, unfortunately, we would not be talking about this insanity if saner heads had prevailed, and today's podcast would then have a different title. The final decision of the Leipzig District Court was to rule in Sony's favor, ordering Quad9 to block all access to CannaPower domains globally. Now, as we all know, Quad9's action of complying with this court order will have zero effect on anyone who does not have 9.9.9.9 configured as their DNS resolver. All of the rest of us - of course it won't have any effect on those people anyway listening to this podcast because we're not going to be pirating music from CannaPower. All of the rest of us can access those CannaPower URLs without any trouble.

So this action by Sony only makes any sense if this is just the first such legal action which Sony intends to use to set a precedent, a legal precedent for other similar actions against other DNS providers. And that's another thing that doesn't really make any sense because the world is also full of DNS providers, many of whom do not care and couldn't care less about making Sony happy. And they're not going to take this action.

What would have made much more sense from a technology standpoint would have been for Sony to contact the responsible and definitely reachable top-level domain provider for the .TO TLD. ".TO" is the Internet country code top-level domain for the Kingdom of Tonga and is administered by the Tonga Network Information Center. And they do pick up the phone. By changing the DNS name server entries for those CannaPower domains, DNS would have disappeared globally once downstream caches expired. But that's not what Sony chose to do.

Five days after this judgment, which came down on March 1st, presumably having recovered from the decision, Quad9 posted their response. They said: "Quad9 has been part of a potentially precedent-setting legal case involving Sony Music. On March 1st, Leipzig Regional Court ruled in favor of Sony Music. This ruling means that Quad9 has no choice but to block the domains in question at a global scale as directed by the court. That said, Quad9 is far from ready to give up the fight in terms of protecting users' access to information.

"Quad9 is shocked that the court ruled in favor of Sony Music, but they are not disheartened and will continue fighting for the freedom of access to information by citizens around the globe. Quad9 feels they were chosen as a target because, as a nonprofit player with a limited budget and small market share, Sony potentially did not

expect Quad9 to have the means of fighting back. This would be an easier means of establishing legal precedent to potentially control domains served by all DNS recursive resolvers.

"Although Quad9 is complying with the ruling in the interim, there are several points that they feel should be brought to the attention of citizens around the globe who value privacy and freedom of access to information on the Internet, as the potential implications of the ruling could reach a global scale. German court decisions are normally limited to Germany, which is why Quad9 has implemented geoIP on its infrastructure in Germany to restrict the domain names in question from being resolved for users querying from Germany. However, there are loopholes, such as VPNs, beyond Quad9's control.

"The court deemed this not to be sufficient. The court's decision ignored the VPN concept and implies that Quad9 must block these domains regardless of how users reach them or from what nation those intentionally disguised queries originate. Quad9 believes this is an exceptionally dangerous precedent that could lead to future global-reaching commercialization and political censorship if DNS blocking is applied globally without geographic limitations to certain jurisdictions.

"The court did not apply the rules of the German Telemedia Act," which sounds a little bit like our Section 230. They said: "Consequently, Quad9 does not enjoy the associated limitations from liability. Quad9 also believes that it should benefit from these exemptions from liability, particularly since the European Lawmakers have noted in the recently adopted Digital Services Act that providers of services establishing and facilitating the underlying logical architecture and proper functioning of the Internet, including technical auxiliary functions, can also benefit from the exemptions from liability, and explicitly mentioned providers of DNS Services." In other words, it sounds like they have a strong case here on appeal.

"The court established that Quad9 accepted the wrongdoer liability. Quad9 feels that this application of the wrongdoer liability is absurdly extreme given the circumstances." And that must make an exception to the German Telemedia Act. "To put this into perspective, applying wrongdoer liability in this setting is akin to charging a pen manufacturer with fraud because a stranger forged documents while using the manufacturer's writing utensil."

**Leo:** Yeah. I agree with that.

**Steve:** Anyway, they go on.

**Leo:** It's very annoying.

**Steve:** I mean, it is really, really wrong. And I wanted to share one additional posting that Quad9 published two days later since it contained some important points and principles about responsibility with DNS. They said: "The Sony Music Entertainment Germany vs. Quad9 Foundation case has brought to light a concerning issue regarding the implementation of blocking measures by DNS recursive resolvers. In addition to concerns around sovereignty and judicial overreach, it is essential to recognize that DNS recursive resolvers are not the appropriate place to implement this type of blocking.

"DNS recursive resolvers play a crucial role in the functioning of the Internet by translating domain names into IP addresses. However, they should not act as

gatekeepers for content, which can be subjective and varies from jurisdiction to jurisdiction. Users of Quad9 opt-in to our service and want the cyber protection that we enable for them. Blocking a domain for distributing malware is motivated by the desire to protect users from malicious or harmful content, such as viruses, phishing scams, or other threats. These types of protective DNS services, blocking a domain from malware, is a form of user opt-in filtering. The intention is to protect users from harm and does not necessarily involve restricting access to specific types of content.

"Blocking a website based on its content is censorship since it involves restricting access to specific information based on someone else's standards or values, rather than allowing individuals to choose what content they wish to access. Most people would like to avoid ransomware on their computers. Still, they might want to browse a website with a specific set of content or an ideological perspective with which we disagree. Blocking a website based on some or all of its content is often motivated by a desire to restrict access to specific types of content.

"Recognizing that DNS recursive resolvers are not the appropriate place to implement content-blocking measures is critical. Arbitrary blocking of content-related domains places an undue burden on DNS recursive resolvers to police online content. DNS recursive resolver operators don't have the legal expertise or access to additional information that might be required to decide what content is legal or illegal."

And finally: "In this ruling, DNS recursive resolvers end up with more culpability and liability for content than social media networks. Concerns around copyright infringement and online piracy are valid, and the courts should address them in appropriate venues. Finding solutions that do not compromise Internet users' security and respect other nations' sovereignty is crucial." And in a one-sentence transparency report that they published, they said: "Quad9's domain blocking policy blocks malicious domains associated with phishing, malware, stalker ware, and command-and-control botnets. Our threat intelligence data comes from trusted cybersecurity partners, not arbitrary corporations or governments. Our policy has been to block malicious domains and not moderate content disputes."

So I am certainly glad that they're going to appeal the decision, and I hope that this ruling will have caught the attention of many other larger entities who share a vested interest in not allowing this single ruling to establish what is clearly the wrong precedent. You know, it's so clear that pursuing legal remedies against the actual publisher of the infringing material, which is what we've always done before, right, I mean, that's what's happened before, that's the right thing to do.

The law should not be about the easy target. It should be about the infringing target. The fact that it may be difficult to hold the infringer to account should not mean that arbitrary and completely unrelated components of Internet infrastructure should become an alternative target. That's just nuts. Although, you know, as I'm saying that, Leo, it occurs to me that we have seen a situation where the copyright holders are going after the downloaders of the content; right? I mean...

**Leo:** Well, that's sensible; right? Those are the ones who are actually infringing.

**Steve:** Exactly. They're obtaining the content. At each end. From the person who is serving it and then the person who's consuming it.

**Leo:** But, yeah, it's hard for them to do that. They tried a bunch of John Doe because it's hard to get the identity; right? They have the IP address. That's all they

have. And they've tried a bunch of John Doe lawsuits. Remember that didn't go very well for the music industry. It turned out to be a black mark for them. Then they went after the ISPs, they still do, with that three strikes thing. It's a long, ugly history. And as far as I can tell, it's accomplished zero. And that's really the main thing to say is it doesn't stop piracy.

**Steve:** It doesn't work.

**Leo:** It just makes it bad for the rest of us. Yeah. Oh, well.

**Steve:** Anyway, they are asking for help and support. I'm hoping that, you know, the EFF and other organizations that have some deep pockets would be able to help these guys.

**Leo:** And I'll promise to Sony I'm never going to download or even listen to an Evanescence album at all, ever. Congratulations. And if you're Evanescence, you might want to look for a new label because they're doing this on your behalf. This is what happened is the artists told the record companies, knock it off. You're punishing our audience.

**Steve:** Our reputation.

**Leo:** And hurting our reputation, exactly. Good stuff. Thank you, Steve. This was a good show. Lot of good information. You could easily have made that Elliptic Curve story the story of the week, but you had two good ones. Two, count 'em, two. And that's why everybody waits with bated breath for Tuesday so they can listen to Security Now!.