

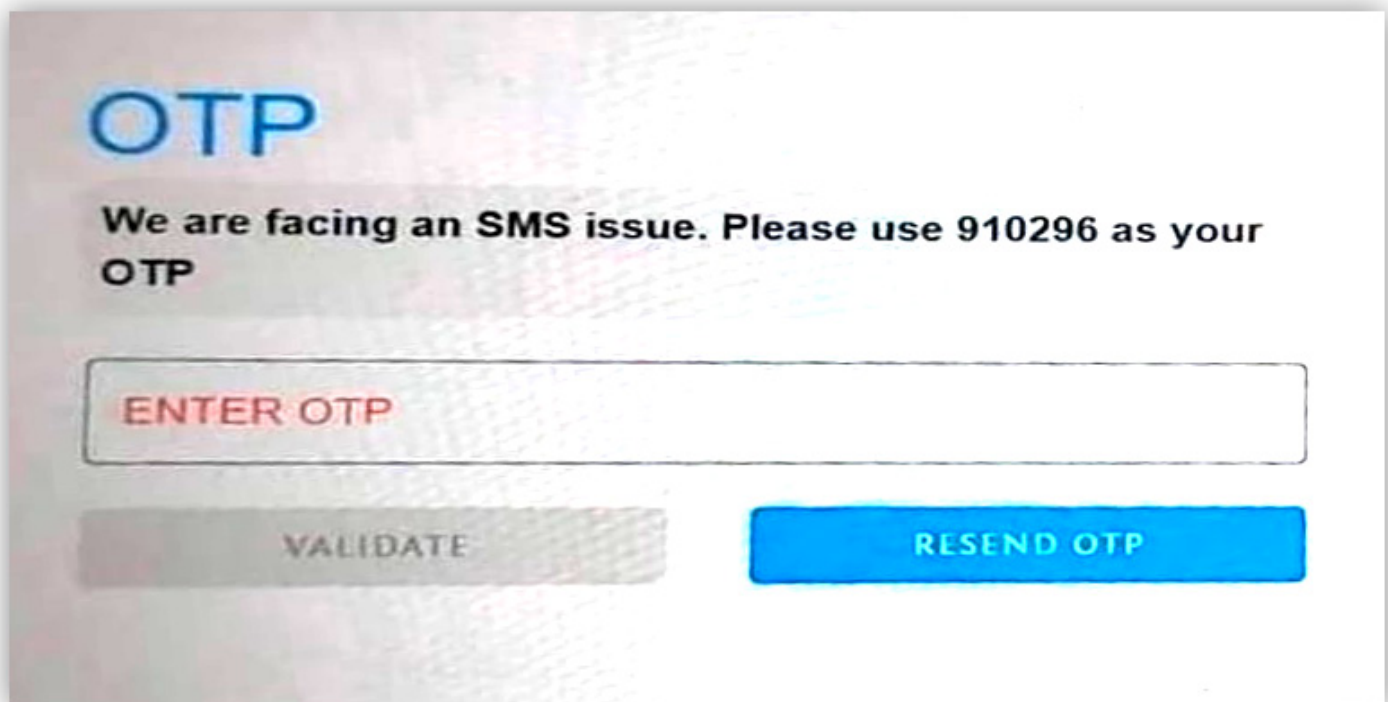
Security Now! #914 - 03-14-23

Sony Sues Quad9

This week on Security Now!

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 #1 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 non-profit 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!

This was their solution?



OTP

We are facing an SMS issue. Please use 910296 as your OTP

VALIDATE

RESEND OTP

Security News

Another Malicious Chrome Extension

After seeing the details of another Chrome extension that was seriously compromising the privacy and the security of more than 160,000 Chrome users, it feels as though we need a better solution for assuring what extensions are doing.

The extension was called "Get cookies.txt" and it was being offered by the official Chrome Web Store. The extension allowed users to export the content of their cookie files in the old Netscape browser cookie format. But concerns about the extension were brought to light in January when a Reddit user discovered that the extension was tracking users by collecting user and browsing data and uploading it to a remote server. 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". Specifically, for every page you visit, it sends:

The page address,

- *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 timezone.*

So that was not good. The URL you're visiting with a unique ID for tracking you, by an extension that has no business whatsoever doing any of that. But then it came to light that the extension was not only performing that bit of tracking, but that it was also proactively sending entire user cookie files back to the extension's publisher. In an update to the Reddit user's initial posting last week, they wrote:

[Update from 2023-03-04: 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 after the extension's upgrade has been in place for some time and many users had obtained the update. After all, what are we constantly telling everyone they need to do? Stay current with all updates!

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 whatever reason on our behalf, this logged-on persistence is accomplished by causing that web browser to accept and store a long-life persistent cookie in that browser's cache. The cookie uniquely identifies you to the site and has the effect of keeping you logged on. 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 the user at any sites where a persistent session cookie may have been set.

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 on.

If by any chance you are one of that extension's 160,000 users, you should seriously consider logging out of any websites which might have you persistently logged in. That will render any stolen cookies useless.

And this takes us back to the question I posed earlier: How do we solve this sort of serious problem? We want our benign browser extensions to be powerful and capable. So they need to be able to do dangerous things – like have access to our global browser cookie cache. But how can we safely trust extensions from unknown authors which might have a hidden agenda? In this case, it appears that the extension started out being much more tame. This allowed it to establish a significant user population and earn ultimately undeserved trust. Over the course of a couple of upgrades its behavior changed significantly as it moved to the dark side.

Today's browser ecosystem doesn't provide any mechanism for deeply vetting an extension's operation. That really is missing. But it would be prohibitively expensive to fully examine every extension. And even if we did, unless every update to that extension was again fully vetted, the same sort of bait-and-switch tactic would work and we would have achieved nothing.

Germany to join the Huawei & ZTE ban

Following the US, 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.

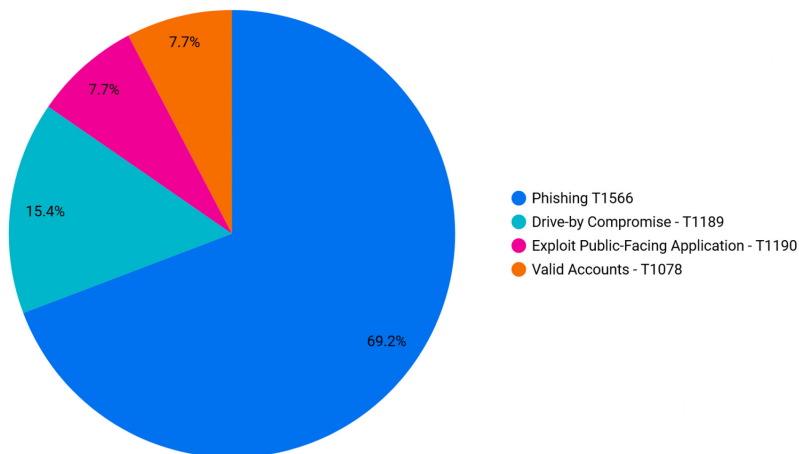
German officials cited the increasingly common fears that Huawei and ZTE equipment could be used for Chinese espionage or sabotage. The German government had previously given the green light for Chinese equipment to be used for its 5G network, but the recent Russian invasion of Ukraine and Russia's attempt to blackmail Germany from aiding Ukraine through its NordStream natural gas pipeline led to a major change of thinking in Berlin. Russia is not China, but the two are 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 US. The collection of countries are already on various schedules to remove any existing Huawei & ZTE equipment from their networks.

I continue to be distressed that this seems xenophobic given the lack of concrete evidence. But there could very well be no evidence despite equipment being compromised. As Congressional expert testimony confirmed, it's virtually impossible to know. 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. We may be seeing a bit of a reflexive reaction to this awareness which will calm down over time.

Putting “phishing” into perspective

The abbreviation DFIR stands for Digital Forensics and Incident Response, and the annual DFIR report was released last week. The report contains interesting statistics and detail, but probably the most interesting fact is the breakdown of the way bad guys are still gaining their initial entry into our systems:



As we see in this overwhelming sea of blue, phishing dominated, with a 69.2% share of intrusion methods where a victim was induced to click a link or open a document which received via either mass or targeted eMail.

The DFIR report noted that as in 2021, in 2022, the majority of intrusions originated from mass email campaigns spreading malware to organizations. They also noted that one of the biggest shifts in this space was the discontinuation of macro attacks in Word and Excel, which was the result of Microsoft’s decision to finally disable macros on files downloaded from the Internet. We covered that decision last year, and just think about that for a moment. For how many years before this did Microsoft's NON-decision to take that simple measure result in untold damage to enterprises using Windows with Word and Excel? Throughout the year years of this podcast we kept lamenting that Microsoft left scripting enabled in eMail despite the fact that no one wanted it. Well, no one but bad guys who kept abusing it to infect users’ machines. The battle to disable eMail scripting was finally won. And now, at long last, we finally have the same thing for Word and Excel macros. But why does each of these sane decisions take so long to finally get made?

Taking second place in “Intrusion entry methods” at 15.4%, is drive-by compromises which include watering hole attacks where a malicious website is set up and victims are lured to visit.

The remaining two ways of gaining entry into networks, which split the remainder at 7.7% each are the exploitation of public facing applications, by which they meant the exploitation of Microsoft Exchange in that case and the abuse of valid accounts, meaning brute forcing RDP or SQL Server to gain an advantage leading to entry.

So there it is. A huge blue slice of pie representing nearly 70% of all intrusions. And similarly preventable. We can hope that at some point someone 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 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? Maybe another 10 years or so before we get that?

The Polynonce attack

The guys at Kudelski Security Research have discovered a new and novel attack, which they call "Polynonce" which enables them, in instances where a weak pseudo-random number generator was used to create a secret signing key, to recover that signing key. This allowed them to, among other things, recover the private keys for hundreds of Bitcoin that was created using weak PRNGs. Here's how the Kudelski researchers explained their work:

In this blog post, we tell a tale of how we discovered a novel attack against ECDSA 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 (we'd be throwing a party instead of writing this blog post), 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.

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. A review of existing literature seemed to confirm that this was indeed a novel insight, so we started digging into it. If you are interested in the math and details surrounding the attack, there's a link to the paper.

We're about to be hearing a lot about ECDSA. ECDSA is an abbreviation for "Elliptic Curve Digital Signature Algorithm." It's 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 have published their matching public key which is 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 now I'm going to read what they wrote, not because I expect anyone to exclaim "Oh, yes, of course, how obvious!" 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. They wrote:

In a nutshell [yeah, right, here comes the "nutshell" version!], 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. If you have a set of N ECDSA signatures (for the same private key) and this recurrence relation is of degree D , then (under some caveats we will talk about later) you can use the ECDSA signature equation to re-write the polynomial in terms of the private key and the recurrence unknown coefficients. 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 ($N = D+3$),

then you can perform a key recovery attack by simply finding roots of a polynomial with known coefficients over a finite field... which is an easy task on a computer!

To run the attack in practice, the following is required: A minimum of 4 signatures generated by the same private key, the associated public key, and the message hash associated with each signature. [All of which will be available anywhere ECDSA is used.] If the nonces obey the recurrence relation, we retrieve the private key used to generate the vulnerable signatures.

The more signatures are used in the attack, the slower it gets, but the more likely it is to succeed: If you attack N signatures and their N nonces follow a recurrence relation of degree at most $N-3$, then you can perform a key recovery attack on ECDSA!

This was their discovery and it's big news because it means that the choice of repeated nonces is truly critical. For any given repeated signer, their chosen nonces, which are used as part of the signing process, must **really** be random... and, by the way, must **absolutely never** be reused.

You can sort of think of it as the price of doing business with ECDSA. 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 reuse a nonce and you sacrifice all of the money in your cryptocurrency wallet.

We tested the attack on a specially crafted set of signatures to verify that it works. It does.

In simpler words, 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 is a catch: nonces are usually output by a pseudorandom number generator (PRNG) 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 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 PRNG what ROT13 is to encryption and "1234" is to secure passwords.

Despite that, due to their simplicity and popularity, they are the default choice for many non-critically secure applications, and it is totally possible that a "placeholder" LCG implementation slips into production code without being replaced by a more secure one.

I'll just interrupt here for a moment to note that LCNG's (linear congruential number generators) come in very handy for many purposes. But in no way imaginable would anyone consider them secure for any purpose. The entire LCNG 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, then add a second constant value. That's it. And that's its attraction. A single multiplication followed by a single addition to produce the next number. So on the negative side it is entirely predictable. Given any output it's possible to immediately produce all future and even all previous outputs. As a source of randomness the term "random" really doesn't apply. What it has going for it 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. Since I think everyone will find this interesting, I'll take a minute to explain...

We hit an unexpected problem with SpinRite, when some very troubled hard disk drives would encounter some damage that they couldn't handle. 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 the error. That would allow SpinRite to zero-in on the one sector that tripped up and drive and work to recover its data.

That what drives are supposed to do. But it turns out that many drives will just hang and wait for SpinRite to timeout the transfer, or they will return a "command aborted" error without any indication of where they were when they became upset. Now we had a problem. 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 64K blocks at a time and switch to single-sector mode to locate any trouble that might be occurring without that block, one sector at a time. That worked great for finding one or more troubled sectors within 127 possible problem sectors. But SpinRite v6.1 achieves its tremendous speed increase by transferring 32,768 sectors at once, which is 16 megabytes. So dropping out of that to switch to one sector at a time mode was no longer an option.

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 16 megabytes 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 zeros-in on that sectors, works on its recovery, reaches a conclusion, then resumes the interrupted transfer with the next sector after the one that caused the interruption.

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 to fill that 16 megabyte 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 requests the same 16 megabyte transfer again. And this time, when the drive craps out without saying where, SpinRite can determine that for itself. It scans forward through the transfer buffer searching for the first sector which matches the noise sector's pattern. THAT will be the first sector in the transfer that was NOT overwritten by the drive's successful reading, so that's where the transfer was interrupted. And that's the sector that SpinRite then needs to work on

recovering. If anyone has been wondering how all of this time is being spent, there's an example. SpinRite v6.1 is going to be a far better SpinRite than has ever existed before.

So that's a perfect example of where a linear congruential number generator can come in handy. We have zero need for security from it. We just want something to produce noise, and we want it fast. Where this should never be used would be for anything remotely approaching security. So the idea that, as these guys suggest, an LCNG might have been left in as placeholder for a real PRNG, and then forgotten... well, that's just horrifying. So, picking up on their investigation:

The Bitcoin blockchain is basically a large, public mine of ECDSA signatures. In fact, 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, 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 time (only approximately, because there is 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 for each signature).

The problem is that these are mainly our speculations, and we have no clue how accurate all these speculations are. So, it's time to verify.

We downloaded and installed Bitcoin Core, the official Bitcoin client, and let it synchronize the whole chain. The sync process took about a day on a fast fiber connection, and the total blockchain size was about 430 GB, up to block 752,759 on September 5, 2022.

Dumping all the signatures and original messages from the raw blockchain data took 24 hours. The resulting output file size was 271 GB and contained 763,020,390 unique signatures. This file contained, on each line: the output_address, ECDSA 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 on. But first, here are some statistics about the dataset.

These signatures were produced by private keys associated with 424,549,744 unique public keys. Of those 424 million public keys, 390 million, or about 92%, produced only 1 signature.

Remember that for Bitcoin, private keys are supposed to be ephemeral and only used once. So it's to be expected that 92% of all signatures would use a completely unique key. That means that no attack is possible here, since these guys are looking for weak nonces occurring during key reuse. So they needed to find instances where multiple signatures used the same key.

There were 34 million public keys with at least 2 signatures, 18 million with at least 3 signatures, 12 million with at least 4 signatures, 9.6 million with at least 5 signatures, and 7.8 million with at least 6 signatures. There was a considerable number of public keys with over 200k signatures. The public key associated with the most signatures had 3.4 million

signatures.

The attack is generic and can be run with at least $N=4$ signatures (linear case) but can also be run with more signatures, for example, 5 signatures for the quadratic case and 6 signatures for the cubic case, or even more signatures. The linear case will also detect repeated nonces but is more general because it can exploit any linear recurrence relation. However, we wanted to go even further and run the quadratic case ($N=5$) because we thought it might give more interesting results. We considered the cost/benefit ratio of performing cubic or higher attacks to not be worthwhile, so we stopped at the quadratic case, meaning batches of 5 signatures. Since we sometimes have more than 5 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 .

So, how did it go?

We ran the sliding window attack with $N=5$ on a 128-core VM, and it was completed in 2 days and 19 hours. The estimated cost of the attack was about \$285 USD.

We broke 762 unique wallets. All of these had a zero balance. 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, it looks like the common mishap of ECDSA implementations using a repeated nonce was the cause of trouble.

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 ECDSA, which is the reuse of a nonce when signing under the same private key.

This meant that a trivial attack against those wallets was possible... and what’s more, somebody had already done that!

Since we only ran the attack using a window of size 5 so far, we may have missed a few vulnerable wallets that would only have been found for public keys that had exactly 4 signatures. So, we re-ran the attack with $N=4$ on only the signatures from wallets with exactly 4 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.

*We suspect (and in some cases have evidence, 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, a value of approximately \$31 million USD at Bitcoin’s peak.*

Okay, so this is as far as I'm going to go once I share their paper's final paragraph:

So, since we aren't sipping Mojitos on a beach in some exotic location, you can tell we didn't gain access to Satoshi's wallet, but 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 ECDSA 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.

So they did, indeed, find and implement an exploit that works on Elliptic Curve DSA when the PRNG used to synthesize the signature's required nonce is not of high quality. The designers of this crypto algorithm have always known of this weakness. Secure crypto absolutely must have high-quality random numbers. So this is a beautiful example of one such exploit.

And during this work they detected and proved that someone had previously discovered the duplication of some nonces, had reverse-engineered those wallets' private keys, broken into those wallets and transferred their bitcoins into their own wallets. All of that was dutifully recorded in the Bitcoin ledger. I have the link to their entire posting for anyone who's interested.

<https://research.kudelskisecurity.com/2023/03/06/polynonce-a-tale-of-a-novel-ecdsa-attack-and-bitcoin-tears/>

So, there are a couple of take-away lessons here. One is that details really matter. 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-dammed certain to only ever use it with truly high quality random nonces. The second takeaway is in the form of a question: If details 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 – such as to protect the storage of hundreds of billions of dollars in cryptocurrency wealth? By the way, for anyone who's curious, as of the beginning of this year, a few months ago, total wealth stored in cryptocurrency was \$804 billion, with around \$320 billion of that in Bitcoin.

And this is sort of the 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 ECDSA 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 some less fragile crypto would probably have been a better choice for something that was to become a crucial global phenomenon. Although, of course, he could have never predicted what was to come.

Plex's RCE now in CISA's KEV

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 just added CVE-2020-5741. What's that you ask? Well, it's a deserialization flaw of untrusted data which was found three years ago in the Plex Media Server for Windows. When exploited, as happened to that unfortunate LastPass developer, after which a distinct lack of fortune was visited upon all LastPass users, a remote, authenticated attacker is then able to execute their arbitrary Python code on the victim's computer.

So we now know that this developer was using a publicly exposed Plex Media Server which was three years out of date, since 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 vulnerability and exploit databases indexed by target vendor. So in the instance of the second LastPass attack, LastPass's developers were identified, probably with the aid of the first attack on the developer network. Then they were tracked down and identified at home and their home IP addresses were scanned. When port 32,400 was found to be accepting inbound TCP connections, 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.

Sci-Fi

"Andor"

I wanted to mention "Andor". In a word... WOW!! (And, of course, there will be no spoilers here, but I want to set the stage and give our listeners a bit of background to make them curious. If you want to hear absolutely nothing in advance then skip forward 60 seconds.)

So... "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 stratified into upper and lower casts.

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 to 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 for new subscribers. I wasn't offered that 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, but neither did it strike me as being all that great. But "Andor" is another thing entirely. I think that this very sober and serious series may be the best Star Wars property I have ever watched. And we cannot wait for season 2.!

Sony Sues Quad9

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, summarized the situation and its danger:

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'm going to dig into much more detail in a moment. But for those of our listeners who were not here for Episode **#12** of this podcast, this is not the first time Sony has earned some negative coverage on this podcast and been named in this podcast's title. Episode #12, dated November 3rd, 2005 ran all of 24 minutes. Its title was: "*Sony's "Rootkit Technology" DRM (copy protection gone bad)*".

That event made quite a splash, and not only with our podcast audience, it earned Sony a permanent page in Wikipedia. In the interest of a bit of nostalgia, and because many of our listeners were not here then, here's the first three paragraphs of Wikipedia's coverage. They write:

A scandal erupted in 2005 regarding Sony BMG's implementation of copy protection measures on about 22 million CDs. When inserted into a computer, the CDs installed one of two pieces of software that provided a form of digital rights management (DRM) by modifying the operating system to interfere with CD copying. Neither program could easily be uninstalled, and they created vulnerabilities that were exploited by unrelated malware. One of the programs would install and "phone home" with reports on the user's private listening habits, even if the user refused its end-user license agreement (EULA), while the other was not mentioned in the EULA at all. Both programs contained code from several pieces of copylefted free software in an apparent infringement of copyright, and configured the operating system to hide the software's existence, leading to both programs being classified as rootkits.

Sony BMG initially denied that the rootkits were harmful. It then released an uninstaller for one of the programs that merely made the program's files visible while also installing additional software that could not be easily removed, collected an email address from the user and introduced further security vulnerabilities.

Following public outcry, government investigations and class-action lawsuits in 2005 and 2006, Sony BMG partially addressed the scandal with consumer settlements, a recall of about 10% of the affected CDs and the suspension of CD copy-protection efforts in early 2007.

Since those early days, Sony has had their troubles. And they do tend to be litigious when they

can find someone to sue. They're happy to throw their weight around without much provocation. Okay. So to the issue of the day.

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 lookup 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 the 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, non-profit foundation headquartered in Zurich, Switzerland, whose sole purpose is improving the privacy and cybersecurity of Internet users. Quad9 has 200 points of presence globally in 90 countries from which they provide these services to individuals and organizations at no cost.

Here's the language from the court which supports the position that Sony's attorneys have described. This is directly from the official court order:

The defendant is ordered to refrain from selling on the territory of the Federal Republic of Germany the music album "Evanescence - The Bitter Truth" 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 domain(s) and to call up there 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:

http://uu.canna.to/links.php?action=popup&kat_id=5&fileid=551125
http://uu.canna.to/links.php?action=popup&kat_id=5&fileid=551499
http://canna.sx/links.php?action=popup&kat_id=5&fileid=551499 and
http://canna-power.to/links.php?action=popup&kat_id=5&fileid=551499

... numeric IP addresses were transmitted, which enabled them to access the hyperlinks provided at the aforementioned addresses to the storage locations of

<http://shareplace.org/?5DF7473B2>
<http://shareplace.org/?0B6DB9EB3>

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 a DNS service provides. Another piece of the official proceedings was also interesting. They describe 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 document describes...

Music content is listed and categorized under the domain www.canna.to – the domain of the website is "CannaPower".

In a letter dated March 23, 2021, [so note that this has been going on for two years] the plaintiff drew the defendant's attention to the infringement and also pointed out the URL. The defendant was requested to put an end to the infringement. The defendant was warned after it failed to remedy the situation. The plaintiff had made every conceivable effort to remove the infringing offer with the involvement of primarily liable parties. The CannaPower website has no imprint. Entries on the domain owner were also not available. Requests for deletion to the host provider remained unanswered. There 2 IP addresses were named. The company InfiumUAB 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 here.

Okay, so in other words, the actual copyright infringers are out of reach of Sony's wrath. From that the court wrote, it doesn't really sound as though Sony tried 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 didn't answer the phone? So that was it?

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 the defendant is liable as a tortfeasor.*" I had to look that one up. The short version 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 is required for users to reach the CannaPower website given its domain name, they – Quad9 – are responsible under civil law for the injury caused to Sony. Oh boy. The court wrote:

According to recent case law, it – meaning Quad9 – is also liable as the perpetrator of a copyright infringement.

Unfortunately, we would not be talking about this insanity at all if saner heads had prevailed and today's podcast would have had 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 the 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 can access those CannaPower URLs without any trouble. So this action by Sony only makes any sense if this is just the first of such legal actions with Sony intending to use this judgement as a precedent for other similar actions against other DNS providers. And that's another thing that doesn't make any sense, because the world is also full of other DNS providers who couldn't care less about making Sony happy.

What would have made much more sense would have been for Sony to contact the top level domain provider for the .TO TLD. “.TO” is the Internet country code top-level domain of the Kingdom of Tonga and is administered by the Tonga Network Information Center. By changing the DNS nameserver entries for those CannaPower domains, DNS would have disappeared globally once downstream caches expired.

Five days later, presumably having recovered from the decision, Quad9 posted their response:

Quad9 has been part of a potentially precedent-setting legal case involving Sony Music. On March 1st, 2023, the Leipzig Regional Court ruled in favor of Sony Music. This ruling means that Quad9 has no choice but to block the domain(s) 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 non-profit 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 prevent 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 commercialized 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, 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.

The court established that Quad9 accepted the wrongdoer liability – Quad9 feels that this application of the wrongdoer liability is absurdly extreme given the circumstances. 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.

Upon the initial injunction in 2021, Quad9 made significant investments to deploy in best-in-class GeoIP technology to prevent users in Germany from accessing the queries in question on Quad9 systems in Germany — However, they were still punished.

Quad9 believes that it is being held to copyright laws that were not intended to be used against recursive resolvers, but instead were meant to apply to services with actual choice and governance over hosting infringed intellectual property or doing business directly with those who house that data. Quad9 does not house any of the data claimed to be infringing and has no business relationship with the organization housing that data. Also, they believe that Sony did not exhaust its means to go after the actual perpetrators or the hosting company first.

Ultimately, Quad9 believes that this case — at its core — is not about copyright infringement but instead a testament to how the jurisdiction of blocking can impact the global ecosystem. They will continue to fight not only for DNS recursive resolvers but also for the freedom of access to information for common global citizens. Also, such a decision may open the floodgates for rights holders to approach other operators of recursive resolvers.

This means tens of thousands of access providers and company network operators, to name but a few. Not all of them will have the courage, expertise, and financial means to properly examine or object to such blocking requests. Instead, they may just block to minimize their legal risk. That will have a chilling effect on freedom and diversity of speech. Let's not forget DNS operators only have a binary choice, namely to block entire domain names or not. If they do, all content and services are no longer accessible, and that may include legal content.

Quad9's first step towards continuing this fight will be appealing the decision reached by the Leipzig Regional Court.

I want to share one additional posting that Quad9 published two days later since it contains some important points and principles about responsibility with DNS:

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 for 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 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 crucial. 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.

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, finally, in a transparency report they published last week, Quad9 included one succinct sentence that sums up their position. They wrote:

Quad9's domain blocking policy blocks malicious domains associated with phishing, malware, stalker ware, and command & 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

I'm glad that they're going to appeal the decision, and I hope that this initial 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.

It's so clear that pursuing legal remedies against the actual publisher of the infringing material is the one 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 alternate target. That's just nuts.

