The End of Entrust Trust

Description: Why does everyone running OpenSSH need to patch immediately? Who just moved 50 bitcoins minted in 2010? (Sadly, it wasn't me.) How are things going with our intrepid Voyager 1? What features have I removed from GRC's email system? And what embarrassingly affordable commercial emailing system do I now recommend without reservation? Who's a "she" and not a "he"? What's recently been happening with Syncthing? Why do I use DNS for freeware release management? And what in the world happened to cause one of the industry's original SSL/TLS certificate authorities to fall from grace and lose all future access to Chrome's root store? Another really great episode of Security Now! is yours for the taking.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-981.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-981-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with some scary news about OpenSSH. An old bug that was fixed is back, and it means a lot of people are going to have to update their OpenSSH. Problems with the speed of Syncthing? Steve has a solution. And then we'll talk about a certificate authority that really seems to have messed it up. It looks like they're in deep trouble. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 981, recorded Tuesday, July 2nd, 2024: The End of Entrust Trust.

It's time for Security Now!, the show where we cover the latest security and privacy news, keep you up to date on what's happening in the world out there with this guy right here, he's in charge, Steve Gibson. Hello, Steve.

Steve Gibson: I'm in charge.

Leo: You are.

Steve: That's right.

Leo: Large and in charge.

Steve: Just don't ask my wife if she thinks I'm in charge.

Leo: No.

Steve: We have today something for everyone. We're going to do, oh, boy, going to take a deep dive into this disastrous OpenSSH problem, which really does mean everybody needs to patch their instances of OpenSSH if it's exposed to the Internet. And I'll explain why, and everyone's going to come away with tinnitus. I mean, really, this one'll wind up your propeller beanies. But we also are going to spend most of our time, like the latter half of the podcast, looking at the politics of the whole question of certificate authority self-management because one of the oldest original certificate authorities in the world, Entrust, has, well, they've gotten themselves in so much trouble that they're no longer going to be trusted by the industry's browsers. So basically all of their SSL/TLS certificate business is gone in a few months.

Leo: Wow.

Steve: It's like, yeah. And so this also gives us an occasion to really look at the behind-the-scenes mechanisms by which this can happen. And of course we've covered CAs falling from grace in the past. This is a biggie. Also, someone just moved 50 bitcoins, minted back in the Satoshi era. Sadly, it wasn't me. But who was it?

Leo: Well, that's interesting. Hmmm.

Steve: Yeah. Also, how are things going with our intrepid Voyager 1 spacecraft? What features have I just removed from GRC's email system? And by the way, just shy of 5,000 people have email from me, from like an hour ago.

Leo: Nice.

Steve: And what embarrassingly affordable commercial emailing system am I now prepared to recommend without reservation, which is what I'm using? And I could not be more impressed with it or its author. Who's a "she" and not a "he," whom I mistakenly referred to last week? What's recently been happening with Syncthing, and what can you do about it? Why do I use DNS for freeware release management, and how? And then we're going to spend the rest of our time taking a look at the title of today's podcast 981 for this July 2nd, "The End of Entrust Trust."

Leo: Wow.

Steve: So I think, you know, a really great episode of Security Now! that is now yours for the taking.

Leo: Well, it will be in a moment. I mean, as soon as I get through my ad it'll be yours for the taking. I'm going to keep it here for a little bit. Wow, very interesting. I can't wait to hear about all of these things. This is going to be a very geeky episode. I like that.

Steve: Yeah.

Leo: I always enjoy it when your shows are a little bit on the propeller-head side. All right. Let's get back to Steve Gibson and your first topic of the day.

Steve: Well, our Picture of the Week.

Leo: Oh, yes.

Steve: I gave this one the caption "Perhaps we deserve to be taken over by the machines." Because...

Leo: Oh, dear. Oh boy, oh boy, oh boy.

Steve: I don't know.

Leo: Describe this for us, will you?

Steve: So we have a close-up photo of a standard intersection traffic light. And permanently mounted on the horizontal member which holds the red, yellow, and green lights is a sign that's got the left turn arrow with a big red, you know, the big circle, red slash through it, clearly indicating that if a police officer is watching you, and you turn left, you'll be seeing some flashing lights behind your car before long. Now, the problem here, the reason I'm thinking, okay, maybe machines will be better at this than we are, is that the signal has a left green arrow illuminated. Meaning, you know...

Leo: Which means turn left.

Steve: Turn left here. So I'm not sure what you do. This is one of those things where the automated driving software in the electric vehicle...

Leo: Oh, imagine.

Steve: It comes here and just quits. It just shuts down and says, okay.

Leo: I don't know what to do.

Steve: I don't know what. I'm seeing that I can't turn left, and the signal is saying I must. And not only that, but it's my turn. So I give up. Anyway. Frankly, I don't know what a human would do if we came to this. It's like...

Leo: Go straight, that's what I'd do. I would not - anything but turn left.

Steve: Wow. Okay. So the big news of the week we're going to start with because we're going to talk about industry politics and management, like self-management of the whole public key certificate mess at the end of the show. But if you survive this first piece, you'll be ready for something as tame as politics. Everyone's buzzing at the moment about this regression flaw. It's a regression because it was fixed back in OpenSSH in 2006. And in a later update it came back. Thus there was a regression to an earlier bad problem. This was discovered by Qualys in OpenSSH, a widely used and inherently publicly exposed service. And in fact, when I say "widely used," we're talking 14 million vulnerable publicly exposed servers identified by Shodan and Censys.

So the developers of OpenSSH have been historically extremely careful. And thank god, because OpenSSH being vulnerable, that would be a big problem. In fact, this is - get a load of this - the first vulnerability to be discovered in nearly 20 years. That's an astonishing track record for a chunk of software. But nevertheless, when you hear that OpenSSH has an unauthenticated remote code execution vulnerability that grants its exploiter full root access to the system with the ability to create a root-level remote shell, affects the default configuration and does not require any interaction from the user over on the server end, that ought to get anyone's attention.

Okay. So we have CVE-2024-6387. So here's what its discoverer, Qualys, had to say. They wrote: "The Qualys Threat Research Unit (TRU) discovered this unauthenticated Remote Code Execution vulnerability in OpenSSH's server" - which is sshd because it's a Linux daemon - "in glibc-based Linux systems. This bug marks the first OpenSSH vulnerability in nearly two decades, an unauthenticated remote code execution that grants full root access. It affects the default configuration and does not require user interaction, posing a significant exploit risk.

"In Qualys TRU's analysis, we identified that this vulnerability is a regression of the previously patched vulnerability which was CVE-2006-5051, reported in of course 2006. A regression in this context means that a flaw, once fixed, has reappeared in a subsequent software release, typically due to changes or updates that inadvertently reintroduce the issue. This incident highlights the crucial role of thorough regression testing to prevent the reintroduction of known vulnerabilities into the environment. This regression was introduced in October of 2020 with OpenSSH 8.5p1." So it's been there for four years. And what this means is any OpenSSH from 8.5p1 on, thus for the last four years, is vulnerable.

Now they say: "OpenSSH is a suite of secure network utilities based on the SSH protocol that are essential for secure communication over unsecured networks. It provides robust encryption, secure file transfers, and remote server management. OpenSSH is widely used on Unix-like systems, including macOS and Linux, and it supports various encryption technologies and enforces robust access controls. Despite a recent vulnerability, OpenSSH maintains a strong security record, exemplifying a defense-in-depth approach and a critical tool for maintaining network communication confidentiality and integrity worldwide."

Okay. So what we're dealing with in this instance is a very subtle and very tight race condition between multiple threads of execution. Remember that not long ago we spent some time looking at race conditions closely. Back then, I used the example of two threads that both wanted to test and conditionally increment the same variable. A race condition fault could occur if one thread first read the variable and tested it, but before it could return the updated value it was preempted, then another thread came along to change that shared variable without the first thread being aware. Then when that first thread had its execution resumed, it would place the updated value back into the variable, destroying whatever that second thread had done. That's a bug, and just something as simple as that can lead to the loss of, you know, lots.

Okay. So today, as it happens, we're going to see a real world example of exactly this sort of problem actually occurring. Okay. So first, though, I want to share Qualys' note about OpenSSH in general. In their technical report about this, they wrote: "OpenSSH is one of the most secure software in the world. This vulnerability is one slip-up in an otherwise near-flawless implementation. Its defense-in-depth design and code are a model and an inspiration, and we thank OpenSSH's developers for their exemplary work."

Then they explain: "This vulnerability is challenging," they write, "to exploit due to its remote race condition nature, requiring multiple attempts for a successful attack. This can cause memory corruption and necessitate overcoming Address Space Layout Randomization (ASLR). Advancements in deep learning may significantly increase the exploitation rate, potentially providing attackers with a substantial advantage in leveraging such security flaws.

"In our experiments" - and I should note that they wrote this, but we'll see later this is one of three sets of experiments, and this was the least worrisome of the three. They wrote: "In our experiments, it takes around 10,000 tries on average to win this race condition. So, for example, with 10 connections being accepted every 600 seconds, it takes on the average of one week to obtain a remote root shell." On the other hand, you can obtain a remote root shell, which is not nothing. And as it turns out, there are ways to optimize this heavily, which we'll get to.

Okay. So of course - and again, they say "around 10,000 tries on average to win the race condition." So that's statistics; right? It could happen on the first try, or never, or anywhere in between. You know, it's like those 50 bitcoin I mined back in 2011. I got lucky. And it's still possible to get lucky today, though it's vastly less likely than it was back then. The great concern is the available inventory, the total inventory of currently vulnerable OpenSSH servers which are publicly exposed to the Internet.

Qualys writes that searches using Censys and Shodan had identified over 14 million potentially vulnerable OpenSSH server instances exposed to the Internet; and that within their own, that is, Qualys's own customer base of users who are using their CSAM 3.0 External Attack Surface Management technology, approximately 700,000 external Internet-facing instances of their own customers are vulnerable. And they explain that this accounts for around 31% of all Internet-facing instances of OpenSSL in their own global customer base. So, you know, of their own customers, they know of 700,000 external Internet-facing instances vulnerable. Censys and Shodan have all of those and an additional 13-plus million more.

Okay. So the way to think about this is that both intensely targeted and diffuse and widespread attacks are going to be highly likely. If a high-value target is running a vulnerable instance of OpenSSH, once this has been fully weaponized, someone can patiently try and retry in a fully automated fashion, patiently knocking at the door until they get in. And what makes this worse, as we'll see, is that the attack is not an obvious flooding style attack that might set off other alarms. Its nature requires a great deal of waiting.

This is why the 10 connections over 600 seconds that Qualys mentioned. Each attack attempt requires, the way it actually works, is a 10-minute timeout. But since 10 can be simultaneously overlapped and running at once against a single target, that brings the average rate down to one completed attack attempt per minute. So on average, you're getting one new connection attempted per minute, which is each of those patiently knocking quietly on the door until it opens for them.

And note that what this means is that a single attacker can be, and almost certainly will be, simultaneously spraying a massive number of overlapping connection attempts

across the Internet. It would make no sense for a single attacking machine to just sit around waiting 10 minutes for a single connection to timeout. Rather, attackers will be launching as many new attempts at many different targets as they can during the 10 minutes they must wait to see whether a single connection attempt succeeded on any one machine.

So to that end, they wrote: "Qualys has developed a working exploit for the regreSSHion vulnerability. As part of the disclosure process, we successfully demonstrated the exploit to the OpenSSH team to assist with their understanding and remediation efforts. We do not release our exploits, as we must allow time for patches to be applied. However, even though the exploit is complex, we believe that other independent researchers will be able to replicate our results."

Okay, and then indeed they detail exactly where the problem lies. I'm going to share two dense paragraphs of techiness, then I'll pause to clarify what they've said. So they wrote: "We discovered a vulnerability, a signal handler race condition, in OpenSSH's server (sshd). If a client does not authenticate within the LoginGraceTime" - which is 120 seconds recently, 600 seconds in older OpenSSH versions - "then sshd's SIGALRM handler is called asynchronously" - that's a key, asynchronously - "but the signal handler calls various functions that are not async-signal-safe." For example, it calls syslog() to log the fact that somebody never authenticated, and it's going to hang up on them.

They said: "This race condition affects sshd in its default configuration. This vulnerability is exploitable remotely on glibc-based Linux systems, where syslog() itself calls async-signal-unsafe functions like malloc() and free()," which allocate and free dynamically allocated memory. They said: "An authenticated remote code execution as root, because it affects sshd's privileged code, which is not sandboxed and runs with full privileges, can result. We've not investigated any other libc or operating system. But OpenBSD is notably not vulnerable because its SIGALRM handler calls syslog_r(), which is an async-signal-safe version of syslog() that was invented by OpenBSD back in 2001."

Okay. So what's going on here is that when someone connects to a vulnerable instance of OpenSSH, as part of the connection management, a connection timeout timer is started. That timer was once set to 600 seconds, which is 10 minutes. But in newer builds giving someone 10 minutes to get themselves connected seemed excessive and unnecessary, so it was shortened to 120 seconds, which is two minutes.

Unfortunately, at the same time they increased the number of simultaneous waiting connections to complete from 10 to 100. So that really did make things worse. And because the attack inherently needs to anticipate the expiration moment, a shorter expiration allows for faster compromise since it's the instant of timer expiration when OpenSSH is briefly vulnerable to exploitation. That window of vulnerability is what the attacker anticipates and exploits. So the more often you get those little windows, the worse off you are.

So upon a new connection the timer is started to give the new connection ample but limited time to get itself authenticated and going. And if the incoming connection just sits there doing nothing, or trying and failing to properly authenticate, regardless of what's going on and why, when that new connection timeout timer expires, OpenSSH drops that still-pending connection. Right? All that makes sense. That's the way you'd want things to operate. Unfortunately, before it does that, as it's doing that, it goes off to do some other things, like make an entry in the system log about this expired connection attempt. So if the wily attacker was doing something on purpose at the precise instant that the connection expiration timer expires, the race condition can be forced to occur.

Leo: Wow.

Steve: Yeah, yeah.

Leo: Modern-day hacks are so subtle and interesting.

Steve: Yeah, because all the easy ones are gone.

Leo: Yeah, that's a good point, yeah. Yeah, right.

Steve: Yeah. The dumb ones we already - we're not doing dumb problems anymore.

Leo: No. Well, you can see how this could have been a regression, too. Now it would easy to reintroduce it.

Steve: Yeah, there was actually an ifdef that got dropped from an update, and that allowed some old code to come back in that had been deliberately deffed out. So just as with the two threads in my original shared variable example, the timer's expiration "asynchronously," and that's the key, it asynchronously interrupts. That means it doesn't ask for permission to interrupt. It just yanks control away from OpenSSH in order to start the process of tearing down this connection that never authenticated itself.

So if the attacker was able to time it right, so that OpenSSH was actively doing something involving memory allocation at the exact time of the timer's expiration, the memory allocations that would then be performed by the timer-driven logging action would conflict and collide with what the attackers were causing OpenSSH to be doing, and that could result in the attacker obtaining remote code execution under full root privilege and actually getting themselves a remote shell onto that machine.

With the massive inventory of 14 million exploitable OpenSSH servers currently available, this is going to be something bad guys will not be able to resist. And unfortunately, as we know, with so many of those forgotten, unattended, not being quickly updated, whatever, there's just no way that attackers will not be working overtime to work out the details of this attack for themselves and get busy.

Qualys explained: "To exploit this vulnerability remotely - to the best of our knowledge, the original exploit of this" - the original vulnerability of this CVE-2006-5051, which I initially mentioned - "was never successfully exploited before," they said, "we immediately face three problems. From a theoretical point of view, we must find a useful code path that, if interrupted at the right time by SIGALRM, leaves sshd in an inconsistent state, and we must then exploit this inconsistent state inside the SIGALRM handler. From a practical point of view, we must find a way to reach this useful code path in sshd, and maximize our chances of interrupting it at the right time. And then from a timing point of view we must find a way to further increase our chances of interrupting this useful code path at the right time, remotely." So theoretical, practical, and timing.

They said: "To focus on these three problems without having to immediately fight against all the modern operating system protections, in particular ASLR and NX," which is execution protection, no execute, they said: "We decided to exploit old OpenSSH versions first, on an x86 system, and then, based on this experience, move to recent versions. So their first experiment was Debian - they showed it as, well, it was the old woody version, which they show as Debian 1:3.4p1-1.woody.3." They said: "This is the

first Debian version that has privilege separation enabled by default and that is patched against all the critical vulnerabilities of that era."

They wrote: "To remotely exploit this version, we interrupt a call to free()" - where memory is being released back to the system - "with SIGALRM, inside sshd's public-key parsing code." Now, that's significant because that means that the attacker is causing OpenSSH to do some public key parsing, probably presenting it with a bogus public key, saying here's my key, use this to authenticate me. So unfortunately, bad guys have been given lots of clues here as a consequence of this disclosure. They know exactly where to look and what to do.

So they said: "We interrupt a call to free() with SIGALRM while sshd is in its public-key parsing code. That leaves the memory heap in an inconsistent state, and exploit this inconsistent state during another call to free(), inside the SIGALRM handler." Probably in syslog.

They said: "In our experiments it takes around 10,000 tries on average to win this race condition," in other words, with 10 connections, which is the MaxStartups setting, accepted per 600 seconds, which is the LoginGraceTime. They said: "It takes around one week on average to obtain a remote root shell." But again, even if you couldn't multiplex this, and you couldn't be attacking a bazillion servers at once, just one attacker camped out on some highly valuable OpenSSH that thinks it's secure because, hey, we use public key certificates. You're never going to guess our password. It's just sitting there, ticking away, knocking patiently at the door an average of once a minute because it can be doing it 10 times over 10 minutes, and eventually the door opens.

Leo: 10,000 tries is hysterical.

Steve: Right. But very patiently; right?

Leo: Because you have to be patient. But still, that's how subtle this race condition is; right?

Steve: Yes, yes. Well, because it also involves vagaries of the Internet timing.

Leo: Right.

Steve: Because you're a remote person, I mean, the good news is the further away you are, if you're in Russia with a flaky network and lots of packet delay, or you're in China, and there's so many hackers that your packets are just competing with all the other attackers, then that's going to introduce a lot more variation. But still, again, this is where patience pays off. You end up with a remote shell with root privilege on the system that was running that server. So the point is, yeah, around 10,000 tries, but massive payoff on the other side.

Okay. Then they said, and I won't go through this in detail, on a newer Debian build where the LoginGraceTime had been reduced from its 600 seconds down to 120, in other words, from five minutes to two minutes, it still took them around 10,000 attempts. But since they only needed to wait two minutes for timer expiration rather than 10 minutes, and they were able to do - oh, no, sorry. On that system they were still only to do 10 at once. So it reduced the wait from five minutes to two minutes, I'm sorry, from 10

minutes to two minutes. They were able to now obtain a remote shell - this is on a newer build - in one to two days, down from around a week.

And finally, on the most current stable Debian version v12.5.0, due to the fact that it has reduced the login time to 120 seconds, but also increased the maximum number of simultaneous login attempts, that so-called "MaxStartups" value, from 10 to 100, they wrote: "In our experiments, it takes around 10,000 tries on average to win this race condition, so on this machine three to four hours with 100 connections accepted per 120 seconds. Ultimately, it takes around six to eight hours on average to obtain a remote root shell, because we can only guess the glibc's address correctly half of the time due to ASLR."

And they finish, explaining: "This research is still a work in progress. We've targeted virtual machines only, not bare-metal servers, on a mostly stable network link with around 10 milliseconds of packet jitter. We are convinced that various aspects of our exploits can be greatly improved, and we've started to work on an amd64, you know, 64-bit world, which is much harder because of the stronger ASLR." And of course that's Address Space Layout Randomization. And the reason 64 bits makes things much worse is that you have many more high bits to allow for more randomized places to locate the code.

And finally, they said: "A few days after we started our work on the amd64, we noticed a bug report in OpenSSH's public Bugzilla, regarding a deadlock in sshd's SIGALRM handler. We therefore decided to contact OpenSSH's developers immediately to let them know that this deadlock is caused by an exploitable vulnerability. We put our amd64 work on hold, and we started to write this advisory."

Okay, so, yikes. We have another new and potentially devastating problem. Everyone running a maintained Linux that's exposing an OpenSSH server to the public Internet, and potentially even major corporations using OpenSSH internally - because, you know, can you trust all your employees? - need to update their builds to incorporate a fix for this immediately. Until that's done, and unless you must have SSH running, it might be worth blocking its port and shutting it down completely.

Leo: I think I have SSH running on my Ubiquiti system and my Synology NAS.

Steve: Yes, probably. And how about on the Synology box?

Leo: Yeah, yeah, yeah. So I'd better check on both of those.

Steve: Yeah.

Leo: And my server, too, come to think of it.

Steve: So, yeah. I mean, no, it's - this is a big deal.

Leo: Yeah.

Steve: So, and as I said at the top, both profiles, you know, a high-value target could be located. And notice that nothing prevents 50 different people from trying to get into the same high-value target at once. So high value is a vulnerability, and just being present is one because the bad guys are going to be spraying the Internet, just looking for opportunistic access. If nothing else, even if they don't care what's going on on your server, they want to put a cryptominer there, or stick a botnet node there. I mean, they're going to want in on these machines where now they have a way. This gives them a way for anything that has been brought up with code for the last four years, since 2020, when this regression occurred. And, you know, they also know some systems will be getting patched, so there's also a rush.

Leo: Right.

Steve: To weaponize this thing and get into the servers they can.

Leo: The way you describe it, it sounds so difficult to implement. But they publish proofs of concept which a script kiddie can implement; right?

Steve: Yup.

Leo: Yeah.

Steve: Yup. It'll end up being productized.

Leo: Right. And you don't need to know what you're doing, you just...

Steve: In the same way that we saw that Windows WiFi bug last week, some guy offering, for five grand you can just buy it right now.

Leo: Wow. Ah, well. What a world.

Steve: Hey, it keeps this podcast full of...

Leo: Thank goodness. That's right. Hello. We appreciate it. Keep up the good work, bad guys. You're keeping us busy. Do you want me to do an ad now, or do you want to keep going? Up to you.

Steve: Yep, perfect timing.

Leo: Okay. You're watching Security Now! with our genius at work, Steve Gibson. Wouldn't be able to talk about this stuff without him, I tell you. He's the key. Now back to Steve Gibson, who is protecting us all week long.

Steve: So a listener of ours, James Tutton, shot me a note asking whether I may have found my 50 bitcoin when he saw an article about 50 bitcoin having been moved from a long dormant wallet. Now, yeah, I wish that was my 50 bitcoin, but I've satisfied myself that they are long gone. But I thought our listeners would enjoy hearing about the general topic of ancient bitcoin movement. The article, which appeared last Thursday at Cryptonews.com, was titled: "Satoshi Era Bitcoin Wallet Awakens."

Leo: Wow. When did you make your 50 bitcoin strike? It was early on.

Steve: February 9th of 2011.

Leo: Okay. So it was one year after this. Oh, that's interesting.

Steve: Yeah, it was early, but it was not this early.

Leo: Okay.

Steve: So they said: "Satoshi Era Bitcoin Wallet Awakens, Moves 50 BTC to Binance." And the Crypto News piece starts out saying: "A Satoshi era Bitcoin wallet address, dormant for 14 years, transferred 50 bitcoin, approximately 3.05 million USD to the Binance exchange on June 27, last Thursday. The wallet is believed to belong to a bitcoin miner who likely earned the 50 bitcoin as mining rewards in 2010."

Leo: This must make you cry.

Steve: Oh, I know, believe me, it's, like, you know. Oh, gosh. It hurts.

Leo: Yeah.

Steve: They said: "On-chain analytics firm Lookonchain revealed the Bitcoin wallet's origins. It's linked to a miner who received 50 bitcoin as a mining reward on July 14, 2010, just months after the Bitcoin network launched." And I'll note that my podcast which Tom Merritt and I did, which was titled "Bitcoin Cryptocurrency," where I explained the operation of the entire Bitcoin cryptocurrency system, how the blockchain works and all that, that aired the following February 9th of 2011.

Leo: Wow, we were really early on that. Wow.

Steve: We were on the ball.

Leo: Yeah.

Steve: So while it's true that solving the bitcoin hash problem way back then resulted in an award of 50 bitcoin, my 50 were different from the 50 that were recently moved. The article continues: "Back in 2010, one bitcoin" - oh, and this explains why I formatted my hard drive - "was valued at a mere $0.003, or 0.3 cents." Right? So, I mean, it was all just...

Leo: A nickel's worth of bitcoin. It wasn't worth worrying about, yeah.

Steve: Well, and remember the faucet, the bitcoin faucet was dripping out bitcoin that anybody could go get for free.

Leo: Right? Free, yeah.

Steve: Right. So they said: "This price was not surpassed until February 2011, reaching $30 by June of that year. Today, Bitcoin trades around $61,000," which, they say, is a 17% drop from its all-time high in mid-March of this year of $73,750 per coin.

Leo: Wow.

Steve: "Satoshi Bitcoin wallets," they write, "which were created during Bitcoin's infancy (2009-2011), hold historical significance. This period marked the time when Bitcoin's enigmatic creator, Satoshi Nakamoto, was still an active presence in the cryptocurrency community. The wallets' historical value, coupled with the limited transactions during that era, makes any movement of funds from them a notable event.

"In 2010, bitcoin mining was accessible to anyone with a personal computer, yielding a reward of 50 bitcoin. This accessibility stands in stark contrast to the current bitcoin mining environment. Four halving events," you know, as in cut in half, "have since reduced the block reward to a mere 3.125 bitcoin." On the other hand, bitcoins are worth 60 grand, so not so mere.

Leo: It gets harder to get a block, to make a block, though, too, because the math is much harder.

Steve: Oh, it's virtually impossible. "These halvings, occurring roughly every four years, are integral to Bitcoin's deflationary model. This recent transfer from a Satoshi Bitcoin wallet is not an isolated incident. It joins a growing list of dormant wallets springing back to life." And of course we know why they're springing. It's because Bitcoin is jumping. So, you know, multiplying the number of bitcoins which were easily earned back then by $60,000 will definitely put a spring in one's step.

Leo: Yeah.

Steve: So they wrote: "In March, a similar event occurred. A miner transferred 50 bitcoin, earned from mining on April 25th, 2010, to Coinbase after 14 years of wallet inactivity. The reactivation of these wallets often stirs interest and speculation within the cryptocurrency community. Many are curious about the intentions behind these moves,

whether they signal a change in market dynamics or simply represent a long-time holder finally deciding to liquidate their assets.

"Bitcoin Whales, individuals or entities holding vast quantities of bitcoin, possess the capacity to influence the cryptocurrency market through their sheer trading volume and holdings. Two such Whale wallets, dormant for a decade, sprang to life on May 12th of 2024, transferring a combined 1,000 bitcoin. On September 12th and 13th of 2013, when Bitcoin was trading at $1.24, each of these two wallets received 500 bitcoin, which was valued at $62K back then.

"In another noteworthy event on May 6th, a Bitcoin Whale moved $43.893 million worth of bitcoin to two wallet addresses. This Whale had remained inactive for over 10 years, having initially received the bitcoin on January 12th, 2014, when it traded at $917."

Leo: This is why it's hard, though, because had you had those 50 bitcoin, when it got to, say, worth $100,000, you would have for sure sold it. You would have said, I'll take the money. That's great. I'm happy.

Steve: Right. And that's why last week's podcast about when is a bad pseudorandom number generator...

Leo: Right, a good thing.

Steve: It kept that guy from decrypting his wallet and selling his bitcoin until, you know, far later, when it became, you know, worth paying some hackers. We don't know what percentage they took, but it would have been nothing if this guy hadn't had them crack his password.

Leo: Many have offered to crack my password. All have failed because it's probably a good password, and it's not - if it's a random password, it's not - and it's done well, which it was, you know, using Bitcoin wallet, it's virtually impossible to brute force.

Steve: Salted and memory hard and slow to do and so forth.

Leo: But someday, you know, I figure this is just a forced savings account. Someday those eight bitcoin will be mine. I don't know, I'll guess the password. Because I must have come up with something I know. Why wouldn't I record it; right?

Steve: I'm sure I know mine.

Leo: Yeah, that's what's sad, yeah.

Steve: But yes, back then we were not fully up to speed on generating passwords at random and having password managers hold onto them.

Leo: Right, right.

Steve: So I could guess my own password.

Leo: I've tried all of the dopey passwords I used by rote back in the day, and none of those worked. So maybe I was smarter.

Steve: And did you rule out monkey123?

Leo: I did, immediately.

Steve: Okay.

Leo: That's the first one I tried. Oh, well. Those eight bitcoins are just going to sit there for a while. That's interesting that 50 is enough to make a news story. That's really...

Steve: Yes. And Leo, there is so much bitcoin that has been lost.

Leo: Oh, yeah. Right.

Steve: I mean, so many people did this.

Leo: Right.

Steve: I'm not a unique hard luck case at all. And besides, I'm doing fine. But a lot of people - and remember when we had some of our listeners come up to us in Boston when we were there for the Boston event. There was one guy in particular who said, "Thank you for that podcast. I retired a long time ago."

Leo: Oh, my gosh. Oh, my gosh.

Steve: "Thanks to listening to the Security Now! podcast. What you said made a lot of sense. I got going. I mined a bunch of bitcoin, and I don't have to work anymore for the rest of my life."

Leo: Wow. That's just good luck. Good fortune.

Steve: Well, yeah.

Leo: Nice.

Steve: Okay. So on the topic of astonishing achievements by mankind, and not cracking your password wallet, I wanted to share a brief update on the status of what has now become the Voyager 1 interstellar probe. NASA's JPL wrote: "NASA's Voyager 1 spacecraft is conducting normal science operations for the first time following a technical issue that arose back in November of 2023. The team partially resolved the issue in April when they prompted the spacecraft to begin returning engineering data, which includes information about the health and status of the spacecraft.

"On May 19th, the mission team executed the second step of that repair process and beamed a command to the spacecraft to begin returning science data. Two of the four science instruments returned to their normal operating modes immediately. Two other instruments required some additional work. But now all four are returning usable science data.

"The four instruments study plasma waves, magnetic fields, and particles. Voyager 1 and Voyager 2 are the only spacecraft to directly sample interstellar space, which is the region outside the heliosphere, the protective bubble of magnetic fields and solar wind created by the Sun.

"While Voyager 1 is back to conducting science, additional minor work is needed to clean up the effects of the issue. Among other tasks, engineers will resynchronize timekeeping software in the spacecraft's three onboard computers so they can execute commands at the right time. The team will also perform maintenance on the digital tape recorder, which records some data for the plasma wave instrument which is sent to Earth twice per year. Most of the Voyagers' science data is beamed directly to Earth, not recorded onboard.

"Voyager 1 now is more than 15 billion miles (24 billion kilometers) from Earth, and Voyager 2 is more than 12 billion miles (20 billion kilometers) from us. The probe will mark 47 years of operations - 47 years of operations - later this year. They're NASA's longest-running and most-distant spacecraft."

Leo: We were young men at the time.

Steve: Yes, Leo.

Leo: Just children.

Steve: We thought we were going to be able to understand all this one day. And, you know, there's more to understand now than there was then.

Leo: Well, that's fun; you know?

Steve: Yeah.

Leo: It's not like everything's a solved problem anymore.

Steve: Nope, we don't have to worry about that.

Leo: No.

Steve: Speaking of solved problems, everything is going well with GRC's email system. And I'm nearly finished with my work on it. The work I'm finishing up is automation for sending the weekly Security Now! email. So I'm able to do it before the podcast while being prevented from making any dumb errors, like forgetting to update the names of links and so forth. I'm about a day or two away from being able to declare that that work is finished. And I should mention, just shy of 5,000 listeners already have the email describing today's podcast with a thumbnail of the show notes that they can click on to get the full-size show notes, a link to the entire show notes text that you and I have, Leo, and then also a bullet-pointed summary of the things we're talking about. So that's all working.

Last week's announcement that I had started sending out weekly podcast summaries generated renewed interest and questions from listeners, both via Twitter or forwarded to me through Sue and Greg. And these were listeners who had apparently been waiting for the news that something was actually being sent before deciding to subscribe to these weekly summary mailings. So now they wanted to know how to do that. All anyone needs to know is that at the top of every page at GRC is a shiny new white envelope labeled "Email subscriptions." Just click that to begin the process. If you follow the instructions presented at each step, a minute or two later you'll be subscribed.

And remember that, if your desire is not to subscribe to any of the lists, but to be able to bypass social media to send email directly to me, you're welcome to leave all of the subscription checkboxes unchecked when you press the "Update Subscriptions" button. That will serve to confirm your email address, which then allows you to send feedback email, Pictures of the Week, suggestions, and whatever else you'd like directly to me by just writing to securitynow@grc.com.

Finally, I wanted to note that the email today's subscribers have already received from me was 100% unmonitored, as I expect all future email will be. So I won't know whether those emails are opened or not. I've also removed all of the link redirections from GRC's email so that clicks are also no longer being counted. This makes the mailings completely blind, but it also makes for cleaner and clearer email. Some of our listeners, as I mentioned last week, were objecting to their clients warning them about being tracked - even though I still don't think that's a fair use of a loaded term when the email has been solicited by the user, and if the notification only comes back to me.

I would never have bothered, frankly, to put any of that in if I'd written the system myself from scratch. But it was all built into the bulk mailing system I purchased. And it is so slick, and it has such lovely graphical displays with pie charts and bar charts and flow charts that it's just - it was so much fun to look at, you know, while it was new. And, frankly, I didn't anticipate the level of backlash that doing this would produce. But then this is not your average crowd, is it. So, you know, we're all Security Now! listeners.

Leo: And, by the way, the average crowd probably knows this, but I will reiterate this. You could go get this PHP program yourself, but the chances are your Internet service provider would immediately block it. You have some sort of special relationship with Level 3 or somebody that allows you to send 5,000 emails out at once. No other Internet service provider would allow that.

Steve: Well, no consumer ISP; right?

Leo: Right, right.

Steve: So anybody who has - any of our people in corporations who have a regular connection to the Internet, you know, not through Cox or through, you know, any of the consumer ISPs.

But anyway, the first two mailings I've done so far, which did contain link monitoring, provided some interesting feedback. For example, three times more people clicked to view the full-size Picture of the Week than clicked to view the show notes. Now, in retrospect that makes sense; right?

Leo: Yes, yes.

Steve: Because most people will be listening to the podcast audio, but they're still curious to see the Picture of the Week, which we have fun describing each week. In any event, I'm over it now. No more single-pixel fetches with its attendant email client freakout, or anything else that might be controversial. What you do with any email you receive from me is entirely up to you. I'm just grateful for everyone's interest.

Leo: There's also an issue with those invisible pixels. Most good email clients, certainly all the ones I use, don't ever load them. They know they're there. They don't warn me. I don't get a warning. They just go, yeah.

Steve: Right. Well, a lot of our listeners do.

Leo: And apparently they do. That's probably Outlook.

Steve: Yeah.

Leo: But, you know, most email clients just go, yeah, sure, yeah.

Steve: Well, anyways, that's all gone. Now, one thing I've been waiting to do, and I've been waiting until I knew I could, was to give a shout-out to the emailing system I chose to use. I've been utterly and totally impressed by its design, its complete feature set, its maturity, and the author's support of his system. And I have to say I feel somewhat embarrassed over what I've received in return for a one-time purchase payment of $169. This thing is worth far more than that.

Now, because I'm me, I insisted upon writing my own subscription management frontend, although I have to say this package's author, a Greek guy whose first name is Panos, and I can't even begin to pronounce his last name because it's about 12 inches long, he has no idea why I've done my own subscription management frontend. He thinks I'm totally nuts because his system, as delivered, does all of that, too. But as Frank Sinatra famously said, "I did it my way." I wanted to, you know, have it look like GRC's pages that our users interacted with.

So nuevoMailer, which is spelled N-U-E-V-O-M-A-I-L-E-R, is an open-source PHP-based email marketing management and mailing solution. It runs beautifully under Windows,

Unix, or Linux. To help anyone who might have any need to create an email facility for their organization or their company or whatever from scratch, or replace one that you're not happy with, I made it this episode's GRC shortcut of the week. So grc.sc/981 will bounce you over to www.nuevomailer.com.

I've had numerous back and forth dialogues with Panos because I've been needing to customize some of the RESTful APIs which his package publishes. I've actually expended his API for my own needs. But, for example, a new feature that's present in the email everyone received from me today for the first time provides a direct link back to everyone's own email subscription management page. So you can click it and immediately be looking at all of the lists and add or remove yourself. To do that I needed to modify some of his code. So I can vouch for the support he provides.

And as I've said, I felt somewhat guilty about paying so little when I've received so much. I mean, this is GRC's email system moving forward forever. So, you know, I'm aware that telling this podcast's listeners about his work, I hope, will likely help him. All I can say is that he deserves every penny he makes. There are thousands, literally thousands of bulk mailing solutions out in the world. This one allows you essentially to roll your own, and I'm very glad I chose it.

Leo: Most people will use something like, what is it, Chimp Mail and/or [crosstalk].

Steve: Mailchimp.

Leo: Mailchimp. Or Constant Contact because they do the mailing, and they've, you know, arranged with whoever's doing their mailing to send out tens of thousands of emails at once. But, yeah, most consumer ISPs won't let you mail anything like that at all.

Steve: No, no, no. In fact, they block port 25, which is SMTP.

Leo: Right. He has a very limited - basically he has a very limited set of possible customers. So you should use it if you can, yeah. Absolutely.

Steve: Okay. A bit of errata, and then we're going to take our next break. Last week's podcast drew heavily on two articles written by Kim Zetter. It's embarrassing that I've been reading, appreciating, and sharing Kim's writing for years, but never stopped to wonder whether Kim would probably associate with the pronoun "he" or "she." Her quite attractive Wikipedia photo strongly suggests that she would opt for "she" - as will I from now on.

Leo: Did you call her "him" last week?

Steve: I think I must have because somebody, you know, said, "Hey, Gibson."

Leo: She's a she, yeah.

Steve: Yeah. What are you talking about here?

Leo: Got to get the pronouns right these days.

Steve: That's right.

Leo: I want to hear what you have to say about Syncthing because I still use it like crazy, and I'm worried now that there's something I should be worried about. But that's, after all, why we listen to the show, isn't it. On we go with the show, Mr. G.

Steve: So I wanted to note that while I'm still a big fan of Syncthing, lately I had been noticing a great deal of slowdown in its synchronization relay servers. I don't think they used to be so slow. I'm unable to get more than 1.5 to 1.8 megabits of traffic through them. While it's not possible to obtain a direct end-to-end - or I should say when it's not possible to obtain a direct end-to-end connection between Syncthing endpoints, an external third-party relay server is required to handle their transit traffic. Everything is super-well encrypted, so that's not the issue. The issue is the performance of this solution.

Since this problem has persisted, or was persisting for me for several weeks, my assumption is that Syncthing's popularity has been growing, and actually we know it has, and is loading down their relay server infrastructure. Which, after all, they just provide for free. No one's paying anything for this. At one point in the past I had arranged for point-to-point connections between my two locations. But some network reconfiguration had broken that. My daytime work location has a machine that runs 24/7. But I shut down my evening location machine at the end of every evening's work. The trouble was that synchronization to that always-on machine had become so slow that I was needing to leave my evening machine running unattended for several hours after I stopped working on it, waiting for my evening's work to trickle out and be synchronized with the machine I'd be using the next morning.

I finally became so - this problem finally became so intolerable that I sat down and punched remote IP filtered holes through my firewalls at each endpoint. Even if pfSense's firewall rules were not able to track public domain names as they are, the public IPs of our cable modems, for example, change so rarely that even statically opening an incoming port to a specific remote public IP is practical. Once I punched those holes, Syncthing was able to make a direct point-to-point connection once again, and my synchronization is virtually instantaneous.

So I just wanted to give a heads-up to anyone who may be seeing the same dramatic slowdown that I was seeing with the use of their relay server infrastructure. It is an amazingly useful free service. And, frankly, helping it to establish direct connections between endpoints also helps to keep the relay servers free, freed up for those who really need them. So that was the issue, Leo, was just the use of a third relay server had recently really ground to a near halt.

Leo: Yeah, I haven't noticed it, but I don't have - you have a much more complicated setup than I do.

Steve: Yeah, I've got like double NAT and all kinds of other crazy stuff going on that really make it a little extra difficult. But for what it's worth, I guess my point is it's worth

taking the time, if you are not seeing a direct WAN, they call it a WAN connection in the UI, with the IP of your remote node, instead you see some mention of relay servers, well, you probably already know how slow things are going.

Leo: Right, right.

Steve: The point is it's worth taking the time to resolve that. And then syncing is just instantaneous.

Leo: Yeah.

Steve: Matt St. Clair Bishop wrote, saying: "Hello, Steve. I've been a listener of Security Now! for some years now. However, as I've edged closer to making my own utilities publicly available, my mind has turned to my method of updating them. I think in my dim and distant memory I remember you saying that you used a simple DNS record to hold the latest edition of each of your public releases, and the client software inspects that record, it being a very simple and efficient mechanism to flag available updates. Could you elaborate at all if you have a spare section in your podcast? I'm personally using C# and the .NET framework as I'm a Windows-only guy. So if you could paint the broad strokes, I should be able to Google the C# detail. SpinRite user, loving all your efforts in this field. Matt St. Clair Bishop."

Okay. So Matt's correct about my use of DNS, and I am pleased with the way that capability has turned out. Anyone who has the ability to look up the IP address, for example, for validrive.rel.grc.com will find that it returns 239.0.0.1. This is because ValiDrive is still at its first release. When I've released an update to ValiDrive, it will be release number two, and I'll change the IP address of ValiDrive.rel, as in release, .grc.com to 239.0.0.2.

Whenever an instance of ValiDrive is launched by any user anywhere in the world, it performs a quick DNS lookup of its own product name "validrive.rel.dns.com" and verifies that the release number returned in the lower byte of the IP address is not higher than its own current release number. If it is, it will notify its user that a newer release exists. What's convenient about this, I mean, there are many things about it, you know, there's no massive flood of queries coming in all over the Internet. It also provides all of its users the anonymity of making a DNS query, as opposed to coming back to GRC, so there's that, too. But this version checking is performed by a simple DNS query packet, and that DNS is distributed and caching.

So it's possible to set a very long cache expiration to allow the cached knowledge of the most recent version of ValiDrive to be spread out across the Internet, varying widely with when each cache expires. This means that when the release number is incremented, the notifications of this event will also be widely distributed in time as those local caches expire. This prevents everyone on the Internet from coming back all at once to get the latest version. And, you know, typically it's not a matter of any urgency.

And to Matt's question and point, I've never encountered a language that did not provide some relatively simple means for making a DNS query. I know that C# and .NET make this trivial. So anyway, that's the story on that.

Oh, and I should mention that 239 is obviously a huge block of IPs which have been set aside. That's the high end of the multicast address space. But the 239 block specifically is

non-routable. So those IPs will never and can never go anywhere. So that's why I chose 239 as the first byte of the IP in the DNS for my release management.

A listener of ours named Carl sent email to me at securitynow@grc.com. He said: "Hi, Steve. Much has been discussed over the recent weeks on your podcast about the upcoming Windows Recall feature and its value proposition versus security and privacy concerns. It has been suggested that the concept started as a productivity assistant that uses AI to index and catalog everything on your screen, and may be more applicable in an office environment than at home.

"However, I think it was just as likely that this concept first started as a productivity monitoring tool, where corporate management can leverage Recall to ensure employees are using their screen time doing real, actual work. Of course, Microsoft realizes they can't possibly market Recall this way, so here we are." He said: "I dread the day Recall is installed on my work computer. Signed, Carl."

Leo: Bad news, Carl. Microsoft already has a product that does that called Viva. They don't need another one. They monitor you all the time, yeah.

Steve: Anyway, Carl's take on this, you know, it aligned with the Evil Empire theory, which as we know I don't subscribe to.

Leo: Right.

Steve: I would say that Recall itself is ethically neutral. You know, it's like the discovery of the chain reaction in the fission of atomic nuclei. That discovery can be used to generate needed power or to make a really big bomb. But the chain reaction itself is just the physics of our universe. Similarly, Recall is just a new capability which could be used to either help or to hurt people. Could employers use it to scroll back through their employees' timeline to see what they've been doing on enterprise-owned machines? That's not yet clear. There are indications that Microsoft is working to make that impossible. But we know that as it was first delivered it would have been entirely possible.

It appears that Microsoft desperately wants to bring Recall to their Windows desktops. It would be incredibly valuable as training material for a local AI assistant and to deeply profile the desktop user as a means for driving advertising selection in a future ad-supported Windows platform. So I suspect they will be doing anything and everything required to make it palatable.

Leo: And as I said, they already have an enterprise product that does that.

Steve: Right, that is deployed, you know...

Leo: In business, yeah.

Steve: Group policy, right.

Leo: Yeah.

Steve: Right. Okay. So this week I want to share the story, and the backstory, of the web browser community again bidding a less than fond farewell to yet another certificate authority. In, as we'll see, what appears to be, or as a result of, as we'll see, what appears to be a demonstration of executive arrogance, Entrust, one of the oldest original certificate authorities, after six years of being pushed, prodded, and encouraged to live up to the responsibilities that accompany the right to essentially print money by charging to encrypt the hash of a blob of bits, the rest of the industry that proactively monitors and manages the behavior of those who have been, dare I say, "entrusted" to do this responsibly, finally reached its limit, and Google announced last Thursday that Chrome would be curtailing its trust of Entrust from its browser's root store.

Okay. So signing and managing certificates is by no means rocket science. There's nothing mysterious or particularly challenging about doing it. It's mostly a clerical activity which must follow a bunch of very clearly spelled out rules about how certificates are formatted and formulated and what information they must contain. These rules govern how the certificates must be managed and what actions those who sign them on behalf of their customers must do when problems arise. And just as significantly, the rules are arrived at and agreed upon collectively.

The entire process is a somewhat amazing model of self-=governance. Everyone gets a say, everyone gets a vote, the rules are adjusted in response to the changing conditions in our changing world, and everyone moves forward under the updated guidance. This means that when someone in this collective misbehaves, they're not pushing back against something that was imposed upon them. They are ignoring the rules that they voted to change and agreed to follow.

"Certificates" have been an early and enduring focus and topic on this podcast because so much of today's security is dependent upon knowing that the entity one is connecting to and negotiating with over the Internet really is who we think they are, and not any form of spoofed forgery. The idea behind a certificate authority is that while we may have no way of directly confirming the identity of an entity we don't know across the Internet, if that entity can provide proof that they have previously, and somewhat recently, proven their identity to a third party, a certificate authority whose identity assertions we do trust, then by extension we can trust that the unknown party is who they say they are when they present a certificate to that effect signed by an authority whom we trust.

That's all this whole certificate thing is about. It's beautiful and elegant in its simplicity. But as the saying goes, "The devil is in the details." And we're going to see today, those who understand the importance of those details can be pretty humorless when they are not only ignored, but flaunted.

The critical key here is that we are completely and solely relying upon a certificate authority's identity assertions, where any failure in such an authority's rigorous verification of the identity of their client customers could have truly widespread and devastating consequences. This is one of the reasons I've always been so impressed with the extreme patience shown by the governing parties of this industry in the face of certificate authority misbehavior. Through the years we've seen many examples where a certificate authority that's trusted really needs to screw up over a period of years, and actively resist improving their game, in order to finally have the industry lower the boom on them. No one wants to do this indiscriminately or casually because it unilaterally puts the wayward CA, the certificate authority, out of the very profitable browser certificating issuing business overnight.

Okay. So what happened? In a remarkable show of prescience, when things were only just heating up, FeistyDuck's "Cryptography & Security Newsletter" posted the following only a few hours before Google finally lowered the boom on Entrust. FeistyDuck wrote: "Entrust, one of the oldest Certification Authorities, is in trouble with Mozilla and other root stores. In the last several years, going back to 2020, there have been multiple persistent technical problems with Entrust's certificates. That's not a big deal when it happens once, or even a couple of times, and when it's handled well. But according to Mozilla and others, it has not been. Over time, frustration grew. Entrust made promises which it then broke. Finally, in May, Mozilla compiled a list of recent issues and asked Entrust to please formally respond.

"Entrust's first response did not go down well, being non-responsive and lacking sufficient detail. Sensing trouble, it later provided another response, with more information. We haven't seen a response back from Mozilla, just ones from various other unhappy members of the community. It's clear that Entrust's case has reached a critical mass of unhappiness." And that's really interesting because this is really the point. All it takes is a critical mass of unhappiness because, as I said, four hours after this was posted, Entrust lost Google. And that's losing the game, essentially, if you're selling certificates for browsers.

So they said: "We haven't heard from other root stores yet. However, at the recent CA/Browser forum meeting, also in May, Google used the opportunity to discuss standards for CA incident response. It's not clear if it's just a coincidence, but Google's presentation uses pretty strong words that sound like a serious warning to Entrust and all other CAs to improve, or else. Looking at the incidents themselves, they're mostly small technical problems of the kind that could have been avoided with standardized validation of certificates just prior to issuance."

And I'll note later that I'll use the term "lint." "Lint" is well understood in the developer community. It means just running a certificate through a lint filter to make sure that there isn't any lint, any debris, any obviously, like, a date set to an impossible number. Or, you know, something obviously missing that the standard says should be there. You know, just do it. But that doesn't happen.

They said: "As it happens, Ballot SC-75 focuses on preissuance certificate linting. If this ballot passes, linting will become mandatory as of March 2025." Meaning it's not there yet. But, boy, after March we're going to see some more booms lowered if people don't lint by default. And that means people are going to have to spend some time and spend some money upgrading their certificate issuing infrastructures. They have not been bothering. Anyway, they said: "It's a good first step. Perhaps the CA/B Forum," you know, the CA/Browser Forum, "will in the future consider encoding the Baseline Requirements into a series of linting rules that can be applied programmatically to always ensure future compliance."

Okay, now, as I noted, a few hours after FeistyDuck posted this, Google made their announcement. Last Thursday, June 27th, the Chrome Root Program and Chrome Security Team posted the following in Google's Security Blog under the title "Sustaining Digital Certificate Security - Entrust Certificate Distrust." And Leo, after taking our final break, I will share what Google wrote and the logic and basically the preamble that led up to this.

Leo: Yeah. As you say, you lose Google, you've pretty much lost the game.

Steve: Game over.

Leo: Game over, man. Back to the saga of Entrust.

Steve: So Google wrote: "The Chrome Security Team prioritizes the security and privacy of Chrome's users, and we are unwilling to compromise on these values. The Chrome Root Program states that CA certificates included in the Chrome Root Store must provide value to Chrome end users that exceeds the risk of their continued inclusion." You should hear a drumbeat in the background here. "It also describes many of the factors we consider significant when CA owners disclose and respond to incidents. When things don't go right, we expect CA owners to commit to meaningful and demonstrable change, resulting in evidenced continuous improvement.

"Over the past few years, publicly disclosed incident reports highlighted a pattern of concerning behavior by Entrust that falls short of the above expectations and has eroded confidence in their competence, reliability, and integrity as a publicly-trusted CA owner. In response to the above concerns and to preserve the integrity of the Web PKI ecosystem, Chrome will take the following actions.

"In Chrome 127 and higher, TLS server authentication certificates validating to the following Entrust roots whose earliest Signed Certificate Timestamp is dated after October 31st, 2024, will no longer be trusted by default." Okay. Then in Chrome's posting they enumerate the exact nine root certificates that Chrome has, until now, trusted to be valid signers of the TLS certificates that remote web servers present to their Chrome browser.

They continue, writing: "TLS server authentication certificates validating to the above set of roots whose earliest Signed Certificate Timestamp is on or before October 31st, 2024, will not be affected by this change. This approach attempts to minimize disruption to existing subscribers using a recently announced Chrome feature to remove default trust based on the SCTs" - that's the Signed Certificate Timestamp, the signing date - "in certificates. Additionally, should a Chrome user or enterprise explicitly trust any of the above certificates on a platform and version of Chrome relying on the Chrome Root Store, the SCT-based constraints described above will be overridden, and certificates will function as they do today. To further minimize risk of disruption, website owners are encouraged to review the Frequently Asked Questions listed below."

Okay. So now - okay. If Chrome were to yank, just summarily yank all nine of those Entrust certs from their root store, at that instant any web servers that were using Entrust TLS certificates would generate those very scary "untrusted certificate warnings" that sometimes we see when someone allows their certificate to expire by mistake. And that makes it quite difficult to use your browser, and most users just say, whoa, I don't know what this red flashing neon thing is, but it's very scary. And if you want to see that, you can go right now to untrusted-root.badssl.com. And there what you will get is a deliberately untrusted certificate so you can see what your browser does: untrusted-root.badssl.com.

Okay. Instead of doing that, Chrome is now able to keep those, I guess I would call them semi-trusted or time-base trusted root certificates in their root store in order to continue trusting any certificates Entrust previously signed and will sign during the next four months, July, August, September, and October, Halloween being the end of that. No Entrust certificate signed from November on will be accepted by Chrome. So that's good. That allows Entrust four months to wind down their services, to decide maybe make a deal with some other CA to, like, purchase their existing customers and transfer them over. I would imagine that's what they'll do.

But there could be no question that this will be a devastating blow for Entrust. Not only will this shut down completely their TLS certificate business, but CAs obtain a great deal

of additional revenue by providing their customers with many related services. Entrust will lose all of that, too.

And of course there's the significant reputational damage that accompanies this which, you know, makes a bit of a mockery of their own name. And there's really nothing they can say or do at this point. The system of revoking CA trust operates with such care to give misbehaving CAs every opportunity to fix their troubles that any CA must be flagrant in their misbehavior for this to occur. As long-time listeners of this podcast know, I'm not of the belief that firing someone who missteps always makes sense. Mistakes happen, and valuable lessons can be learned. But from what I've seen, and what I'm going to share, I'll be surprised if this is a survivable event for Entrust's Director of Certificate Services, a guy named Bruce Morton.

Way back in 1994, Entrust built and sold the first commercially available public key infrastructure. They started all this. Five years later, in 1999, they entered the public SSL market by chaining to the Thawte Root and created Entrust.net. And as I said, their name has been around forever. You know, I've seen it when I've looked at lists of certificates. There's Entrust. Ten years later, Entrust was acquired for $124 million by Thoma Bravo, a U.S.-based private equity firm.

Leo: Mm-hmm. There you have it.

Steve: Uh-huh.

Leo: In a nutshell. Add this one to the list. Wow.

Steve: Uh-huh. I don't know, and I'm not saying, whether being owned by private equity may have contributed to their behavior and their downfall. But, if so, they would have that in common with LastPass.

Leo: Yeah.

Steve: As you said, Leo...

Leo: And Red Lobster. And about a million other companies in the United States in the last 10 years that have been bought by private equity and then drained of their resources for money. It's sad.

Steve: Yup. Google, in their FAQ, answering the question "Why is Chrome taking action?," replied: "Certification Authorities serve a privileged and trusted role on the Internet that underpin encrypted connections between browsers and websites. With this tremendous responsibility comes an expectation of adhering to reasonable and consensus-driven security and compliance expectations, including those defined by the CA/Browser TLS Baseline Requirements.

"Over the past six years, we have observed a pattern of compliance failures, unmet improvement commitments, and the absence of tangible, measurable progress in response to publicly disclosed incident reports. When these factors are considered in aggregate and considered against the inherent risk each publicly-trusted CA poses to the

Internet ecosystem, it is our opinion that Chrome's continued trust in Entrust is no longer justified."

And, okay, this makes a key point: It's not any one thing that Entrust did, taken in isolation, that resulted in this loss of trust. The loss of trust resulted from multiple years of demonstrated uncaring about following the rules that they had voted upon and agreed to as a member of this group. No one wants to make Entrust an example. Too many lives will be negatively impacted by this decision. But the entire system only functions when everyone follows the rules they've agreed to. Entrust refused to do that, so they had to go. Let's take a look at some specifics.

For example, a few months ago, following an alert from Google's Ryan Dickson, Entrust discovered that all of its EV certificates issued since the implementation of changes due to Ballot SC-62v2, which amounted to approximately 26,668 certificates, were missing their CPS URIs, in violation of the EV Guidelines. Entrust said this was due to discrepancies and misinterpretations between the CA/Browser Forum's TLS Baseline Requirements and the Extended Validation Guidelines. Entrust chose to not stop issuing the EV certificates - that's a violation of the rules - and did not begin the process of revoking the mis-issued certificates - that's another violation.

Instead, they argued that the absence of the CPS URI in their EV certificates was due to ambiguities in CA/B Forum requirements, which was not the case. They said that the absence of the CPS URI had no security impact - that's arguably true - and that halting and revoking the certificates would negatively impact customers and the broader Web PKI ecosystem. In other words, they thought they were bigger than the rules, that the rules were dumb, or that the rules didn't apply to them. Everyone else has to follow them, but not them. Entrust then also proposed a ballot to adjust the EV Guidelines so that they would not be out of compliance, to not require the CPS URI. They also argued that their efforts were better spent focusing on improving automation and handling of certificates, rather than on revocation and reissuance. Wow.

Okay, now, the CPS URI is truly incidental. CPS stands for Certification Practice Statement, and EV certs are now supposed to contain a CPS URI link pointing to the CA's issuing document. So is leaving that out a big deal? Probably not from a security standpoint. But it's worrisome when a CA intentionally defies the standards that everyone has agreed to follow, and then argues about them, and is deliberately, knowingly, in misissuance.

A security and software engineer by the name of Amir Omidi has worked on maintaining certificate issuance systems at Let's Encrypt and Google Trust Services, and he's very active in the PKI space. His GitHub account contains 262 repositories, and it appears that he's currently working on a project named "boulder" which is an ACME-based certificate authority, written in Go. And before that was "zlint," an X.509 Certificate Linter focused on Web PKI standards and requirements.

Yesterday, just Monday, yesterday he posted a terrific summary of the way the Public Key Infrastructure industry thinks about these things. He wrote: "Entrust did not have one big explosive incident. The current focus on Entrust started with this incident. On its surface, this incident was a simple misunderstanding. This incident happened because up until the SC-62v2 ballot, the CPS URI field in the certificate policy extension was allowed to appear on certificates. This ballot changed the rules and made this field be considered 'not recommended.' However, this ballot only changed the baseline requirements and did not make any stipulation on how Extended Validation certificates must be formed. The EV guidelines still contained rules requiring the CPS URI extension.

"When a CA," writes Amir, "when a CA has an incident like this, the response is simple: Stop misissuance immediately. Fix the certificate profile so you can resume issuance. In

parallel, figure out how you ended up missing this rule and what the root cause of missing this rule was. Revoke the misissued certificates within 120 hours of learning about the incident. Provide action items that a reasonable person would read and agree that these actions would prevent an incident like this happening again." In other words, this is all understood. Entrust ignored it.

He writes: "When I asked Entrust if they've stopped issuances yet, they said they haven't, and they don't plan to stop issuance. This is where Entrust decided to go from an accidental incident to willful misissuance. This distinction is an important one," he says. "Entrust had started knowingly misissuing certificates. Entrust received a lot of pushback from the community over this. This is a line that a CA shouldn't, under any circumstances, cross. Entrust continued to simply not give a crap" - and I changed that word to be a little more politically correct - "even after Ben Wilson of the Mozilla Root Program chimed in and said that what Entrust is doing is not acceptable." And then he writes: "Entrust only started taking action after Ryan Dickson of the Google Chrome Root Program also chimed in to say this is unacceptable."

I'll interrupt to mention that this is an important distinction. The executives at Entrust appeared not to care about any of this until Google weighed in with the power of their Chrome browser. That was a monumental mistake, and it demonstrated a fundamental misunderstanding of the way the CA/Browser forum members operate. None of this operates on the basis of market power. It's only about agreeing to and then following the rules. It's not about, "Oh yeah? Make me!" We're not in the schoolyard anymore.

Amir continues: "Entrust's delayed response to the initial incident, spanning over a week, compounded the problem by creating a secondary 'failure to revoke on time' incident. As these issues unfolded, a flurry of questions arose from the community. Entrust's responses were often evasive or minimal, further exacerbating the situation. This pattern of behavior proved increasingly frustrating, prompting me to delve deeper into Entrust's past performance and prior commitments.

"In one of my earlier posts, I found that Entrust had made the promise that, 'One, we will not make the decision not to revoke,' which they just had. 'We will plan to revoke within 24 hours or five days as applicable for the incident,' which they've said they won't. 'We will provide notice to our customers of our obligations to revoke and recommend action within 24 hours or five days based on the Baseline Requirements,' which they won't do because they're not going to revoke in the first place." He says: "This pattern of behavior led to a troubling cycle: Entrust making promises, breaking them, and then making new promises, only to break those, as well.

"As this unfolded, Entrust and the community uncovered an alarming number of operational mistakes, culminating in a record 18 incidents within just four months. Notably, about half of these incidents involved Entrust offering various excuses for failing to meet the 120-hour certificate revocation deadline - ironically, a requirement they had voted to implement themselves." He said: "I do want to highlight that the number of incidents is not necessarily an indication of CA quality. The worst CA is the CA that has no incidents, as it's generally indicative that they're either not self-reporting, or not even aware that they're misissuing."

Okay. So in other words, mistakes happen. Everyone understands that. No one needs to be perfect here. But it's how the mistakes that are discovered are then handled that demonstrates the trustworthiness of the CA.

Amir said: "Due to the sheer number of incidents, and Entrust's poor responses up until this point, Mozilla then asks Entrust to provide a detailed report of these recent incidents. Mozilla specifically asks Entrust to provide information regarding" - and we have some bullet points. "The factors and root causes that led to the initial incidents, including

commonalities among the incidents, and any systemic failures." Okay, now, listen to this, I mean, because this is really Mozilla getting up in Entrust business. And Entrust apparently doesn't take kindly to that.

Okay. So literally, Mozilla confronts Entrust and says we want to know the factors and root causes that led to the initial incidents, highlighting their commonalities among the incidents and any systemic failures. We want to know Entrust's initial incident handling and decision-making in response to these incidents, including any internal policies or protocols used by Entrust to guide their response and an evaluation of whether their decisions and overall response complied with Entrust's policies. We want your practice statement, and the requirements of the Mozilla Root Program.

In other words, explain to us, and we're not kidding here, how this happened. Like, are you ignoring our own policies, or are these your policies? In other words, WTF? And we want it in detail, please. We need also - I mean, literally, this is in his letter - a detailed timeline of the remediation process and an apportionment of delays to root causes. So please, you know, elaborate on the delays which were involved in this because, you know, we're out here. We don't understand. Also, an evaluation of how these recent issues compare to the historical issues referenced above and Entrust's compliance with its previously stated commitments, which everyone already knows is missing.

"Mozilla also asked," writes Amir, "that the proposals meet the following requirements." So literally, these are what we need to know, and here are the requirements you must meet in your reply. "We want clear and concrete steps that Entrust proposes to take to address the root causes of these incidents and delayed remediation. We want measurable and objective criteria for Mozilla and the community to evaluate Entrust's progress in deploying these solutions. And we want a timeline for which Entrust will commit to meeting these criteria."

As Amir said, even here, he said: "Mozilla gave Entrust a one-month deadline to complete this report. Mozilla's email served a dual purpose. It was both a warning to Entrust and an olive branch, offering a path back to proper compliance. This presented Entrust with a significant opportunity. They could have used this moment to demonstrate to the world their understanding that CA rules are crucial for maintaining Internet security and safety, and that adhering to these rules is a fundamental responsibility. Moreover, Entrust could have seized this chance to address the community, explaining any misunderstandings in the initial assessment of these incidents and outlining a concrete plan to avoid future revocation delays.

"Unfortunately, Entrust totally dropped the ball on this. Their first report was a rehash of what was already on Bugzilla, offering nothing new. Unsurprisingly, this prompted a flood of questions from the community. Entrust's response? They decided to take another crack at it with a second report. They submitted this new report a full two weeks after the initial deadline.

"In their second report, Entrust significantly changed their tone, adopting a more apologetic stance regarding the incidents. However, this shift in rhetoric was not matched by their actions. While expressing regret, Entrust was still overlooking certain incidents, delaying the revocations of existing misissuances, and failing to provide concrete plans to prevent future delayed revocations. An analysis of these 18 incidents and Entrust's responses serves as a prime example of mishandled public communications during a crisis."

Okay, now, stepping back from this for a moment, the only way to really read and understand this is that the executives at Entrust - and yes, Leo, a public equity owned firm.

Leo: A private equity, yeah...

Steve: A private equity, sorry, private equity...

Leo: Thoma Bravo.

Steve: ...owned firm, yeah, the executives at Entrust didn't really take any of this seriously. They acted as though they were annoyed by the gnats buzzing around them who were telling them how they should act and what they should do.

Amir says: "The consensus among many community members is that Entrust will always prioritize their certificate subscribers over their obligations as a Certificate Authority." And there it is in a single sentence. "The consensus among many community members is that Entrust will always prioritize their certificate subscribers over their obligations as a Certificate Authority."

And he said: "This practice fundamentally undermines Internet security for everyone. Left unchecked, it creates a dangerous financial incentive for other CAs to ignore rules when convenient, simply to avoid the uncomfortable task of explaining to subscribers why their certificates need replacement. Naturally, customers prefer CAs that won't disrupt their operations during a certificate's lifetime. However, for CAs that properly adhere to the rules, this is an impossible guarantee to make." In other words, no one should expect CAs to be perfect. The community here doesn't. They understand mistakes will happen. But it's maintaining the integrity of the system is more important than anything else.

He says: "Furthermore, these incidents were not new to Entrust. As I've covered in earlier posts, Entrust has continuously demonstrated that they're unable to complete a mass revocation event in the 120 hours defined by and required by the baseline requirements. This pattern of behavior suggests a systemic issue rather than isolated incidents. Despite there being over a dozen root programs, there are only four that are existentially important for a Certificate Authority: The Mozilla Root Program, used by Firefox and practically all Linux distribution and FOSS software. The Chrome Root Program, used by Chrome - the Browser and the OS - and some Androids. The Apple Root Program, used by everything Apple. And the Microsoft Root Program used by everything Microsoft." He finishes: "Enforcement over the operational rules of a CA has been a struggle in the past. A root program only has a binary choice, to either trust or distrust a certificate authority.

Now, there's one last much shorter piece of interaction that I want to share. It was written by Watson Ladd, who studied math at Berkeley and is presently a principal software engineer at Akamai. Among his other accomplishments, he's the author of RFC 9382 which specifies SPAKE2, a password-authenticated key exchange system, and his name has been on about six other RFCs. So, you know, he's a techie, and he's in the game.

In the public discussion thread about Entrust's repeated and continuing failings to correct their mistakes and live up to the commitments they had made to the CA/Browser community, Watson publicly addressed a note to Bruce Morton, Entrust's Director of Certificate Services, who has been the face of Entrust's repeated failures, excuses, and defiance.

Watson Ladd wrote: "Dear Bruce. This report is completely unsatisfactory. It starts by presuming that the problem is four incidents. Entrust is always under an obligation to

explain the root causes of incidents and what it is doing to avoid them as per the CCADB incident report guidelines. That's not the reason Ben and the community need this report." And here he's referring to Mozilla's Ben Wilson, who initially asked Entrust to explain how they would deal with these ongoing problems and demonstrate how they would be prevented in the future. As we know, Entrust's Bruce Morton basically blew him off, apparently because he wasn't from Google.

Anyway, Watson says: "That's not the reason Ben and the community need this report. Rather, it's to go beyond the incident report to draw broader lessons and to say more to help us judge Entrust's continued ability to stay in the root store. The report falls short of what was asked for, in a way that makes me suspect that Entrust is organizationally incapable of reading a document, understanding it, and ensuring each of the clearly worded requests is followed."

Leo: Wow.

Steve: Yeah. The implications for being a CA are obvious.

Leo: He's mad. Holy cow.

Steve: They are, they all are. He said: "To start, Ben specifically asked for an analysis involving the historical run of issues and a comparison. I don't see that in this report at all. The list of incidents only has ones from 2024 listed. There's no discussion of the two issues specifically listed by Ben in his message.

"Secondly, the remedial actions seem to be largely copy and pasted from incident to incident without a lot of explanation. Saying the organizational structure will be changed to enhance support, governance, and resourcing really doesn't leave us with a lot of ability to judge success or explain how the changes made - sparse on details - will lead to improvements. Similarly, process weaknesses are not really discussed in ways that make clear what happened. How can I use this report if I was a different CA to examine my organization and see if I can do better? How can we, as a community, judge the adequacy of the remedial actions in this report?

"Section 2.4 I find mystifying. To my mind there's no inherent connection between a failure to update public information in a place where it appears, a delay in reconfiguring a responder, and a bug in the CRL generation process beyond the organizational. These are three separate functions of rather different complexity. If there's a similarity, it's between the latter two issues where there was a failure to notice a change in requirements that required action, but that's not what the report says. Why were these three grouped together, and not others? What's the common failure here that doesn't exist with the other incidents?

"If this is the best Entrust can do, why should we expect Entrust to be worthy of inclusion in the future? To be clear, there are CAs that have come back from profound failures of governance and judgment. But the first step in that process has been a full and honest accounting of what their failures have been, in a way that has helped others understand where the risks are and helps the community understand why they are trustworthy. Sincerely, Watson Ladd."

Leo: Watson was hopped up. Doesn't sound like it, but that's what an engineer sounds like when they get really mad.

Steve: Well, now, Leo.

Leo: Yes.

Steve: I don't know these Entrust guys at all. But given the condescension they've exhibited, it's not difficult to picture them as some C Suite stuffed shirts who have no intention of being "judged by" and pushed around by a bunch of pencil-necked geeks. But, boy, did they read this one wrong. Those pencil-necked geeks with their pocket protectors reached consensus and pulled their plug, ejecting them from the web certificate CA business they had a hand in pioneering.

Leo: This is what happens when people who only run businesses don't understand the difference between a business, a profit-seeking enterprise, and a public trust; right?

Steve: Yes.

Leo: And they don't understand that in order to run your business, you have to satisfy the public trust part. You can't just say, yeah, yeah, whatever. You've got to respond.

Steve: And notice that Entrust was taken private.

Leo: Yeah.

Steve: So they no longer had, literally, a public trust.

Leo: Of their business, except that a certificate authority has a public trust, sorry. That's the job.

Steve: Yes, it is a public trust.

Leo: It's a public trust.

Steve: Yes.

Leo: Wow. Clearly they were in over their heads or something.

Steve: Well, but they started this business. I mean...

Leo: Is it the same people, though, really?

Steve: Well, that's exactly - that's a great question.

Leo: Yeah. It's like Joe Siegrist wasn't at LastPass at the end.

Steve: Yes, exactly, exactly. So some middle managers rotated in and didn't understand that if they - that the aggravation that was simmering away in this CA/Browser Forum was actually capable of boiling over and ending their business.

Leo: Yeah. They didn't get it. They really didn't get it. And you know what? You know, well, they know now. Whoops.

Steve: Yeah. It's over. They appear to have believed that the rules they agreed to did not apply to them. Or, you know, I thought maybe it was the extreme leniency that the industry had been showing them that led them to believe that their failures would never catch up with them. And, boy, but the worst thing that they did was just to basically blow off the heads of these root programs when they said, hey, look, we see some problems here. We need you to convince us that you're worthy of our trust.

Leo: Yeah.

Steve: And the Entrust people probably just said "Eff you." Well, now they're going to be out of business in four months.

Leo: So without the trust of Chrome, and I presume other browsers, well, Mozilla's clearly going to...

Steve: Oh, Mozilla will be right on their heels, yes.

Leo: And then Edge and everybody else who does certificates will follow; right? But it doesn't matter. If Chrome doesn't - if you're not in Chrome...

Steve: That's right.

Leo: People using Chrome won't be able to go to sites that have Entrust certificates. Game over. Right?

Steve: Yes.

Leo: Yeah.

Steve: Yes. Entrust is out of business. The only thing they could do would be to sell their business. Remember when Symantec screwed up royally?

Leo: Oh, it happens all the time.

Steve: They ended up having to - they had to sell their business to DigiCert.

Leo: Right. Right.

Steve: Because, you know.

Leo: So they also might face the wrath of people who use Chrome, rather websites that use their certificates when their customers can't get to them. I mean, they might be some big companies.

Steve: Well, all the certs that have been issued stay valid.

Leo: Okay.

Steve: That's the good thing. So any site, even through Halloween, even through the end of October - because what Chrome is doing is they're looking at the signing date.

Leo: Ah.

Steve: And they're saying anything Entrust signs after October 31st, 2024, we're not going to trust.

Leo: Right.

Steve: We're going to take the user to the invalid certificate page.

Leo: So there is a little bit of a warning going out to people who use Entrust certificates. You're going to need a new certificate from a different company now.

Steve: Yes, yes.

Leo: You know, by October.

Steve: Yes, yes. You might as well switch. You could buy an Entrust certificate up until Halloween.

Leo: Get you through Halloween.

Steve: And it would stay good for the life of the certificate, which has now been reduced to one year. What is it, 386 days or something.

Leo: That's right, yeah, yeah.

Steve: So it's, you know, time to switch providers. So Entrust loses all of that ongoing revenue from certificate renewals. They also lose all of the second order business that their customers got. You know, they're like, oh, well, we'll also sell you some of this and some of that. And you probably need some of this over here. I mean, they're in a world of hurt. And, you know, unfortunately, it couldn't happen to a nicer group of guys because they entirely brought this on themselves. It was literally their arrogance.

Leo: Yeah.

Steve: You know, we are not going to do what you are asking us to do.

Leo: Right.

Steve: And so the consensus was, okay.

Leo: You don't have to, but we don't have to trust you.

Steve: We're not making you. We're just going to stop believing you.

Leo: Right. That's really interesting. Usually what happens with private equity is they buy a company and sell off assets or somehow make money. The example of Red Lobster comes to mind. The private equity company that bought the Red Lobster restaurant chain took the real estate that was owned by all the restaurants and sold it to a separate division, making almost enough money to compensate for the debt they'd incurred to buy it because that's what happens. They borrow money. Then they squeeze the company, get all the debt paid off, and then - but the problem was now the restaurants had to pay rent, and couldn't, and they went out of business. And that's what happens. You squeeze, get your money back, get your money out, and then you don't care what happens.

So I don't know what they were able to squeeze out of Entrust, but it may be they got what they wanted out of it, and they don't care at this point. That's what it feels like. They just don't care. They could come back from this; right? They could say, oh, yeah, wait a minute, oh, sorry, we were wrong. Could they? Or is it too late?

Steve: I don't think so. I mean, maybe they'd change their name to Retrust.

Leo: Wow.

Steve: No, I mean, it's over. I mean, they can't, like, say oh, oh, oh, we're really sorry, we didn't understand. I mean, I've never seen any of this ever reversed. These are slow-moving icebergs. And when the iceberg hits your ship, you know, it doesn't turn around.

Leo: It rends you from stem to stern.

Steve: It does.

Leo: OutofSync in our Discord says that Entrust is about 0.1% of all certs, compared to Let's Encrypt, which is now 53% of all certs. Why not? It's free; right? And if you've got to renew it every 384 days, you might as well just go with Let's Encrypt.

Steve: Yeah, last year we talked about the big seven, where if you only trusted seven CAs, you've got 99.95 or something. And Entrust is not one of them.

Leo: I don't remember Entrust. It wasn't on that list. Okay. So maybe this is just incompetence or something.

Steve: Well, it's - yeah. That, I mean, you're incompetent if you're a manager who doesn't know how to read email and understand the importance of it to your career. And I doubt this Bruce guy is going to have a job in four months.

Leo: It won't be, let's put it this way, it will not be in the certificate business.

Steve: It cannot be in the certificate business.

Leo: Steve Gibson, I love it. This was a fun one. A little scary early on with OpenSSH, but you kind of redeemed it with some humor towards the end. What a story that is. Amazing. Steve's at GRC.com. Now, if you go to GRC.com/email, you can sign up for his mailing lists. You don't have to, though. All it will do is validate your email so that you can then email him. So it's the best way to be in touch with Steve.

If you're there, you should check out SpinRite. Version 6.1 is out. It is now easily the premier mass storage performance enhancer. It's like Viagra for your hard drive. No. Performance enhancer, maintenance and recovery utility.

Steve: Well, you want your hard drive to be hard.

Leo: You do. You don't want a floppy drive. We're not talking floppies here. This is a hard drive. Or an SSD. Let's be fair, works on SSDs, as well.

Steve: Bye.