Rethinking Revocation

Description: A million domains are vulnerable to the "Sitting Duck" attack. What is it? Is it new? Why does it happen? And who needs to worry about it? A CVSS 9.8 (serious) remote code execution vulnerability has been discovered in Windows' RDL (Remote Desktop Licensing) service. Patch it before the bad guys use it! All of AMD's chips have a critical (but patchable) microcode bug that allows boot-time security to be compromised. Now what? Microsoft apparently decides not to fix a simple Windows bug that allows anyone to easily crash Windows with a Blue Screen of Death anytime they wish. You sure don't want that in your Windows startup folder! GRC's IsBootSecure freeware is updated and very nearly finished. Believe it or not, the entire certificate revocation system that the industry has just spent the past 10 years getting working is about to be scrapped in favor of what never worked before. Go figure.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-987.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-987-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. I am here in my new attic studio. But we have lots to talk about, including a new attack called Sitting Duck. Are you a sitting duck? Why all AMD, almost all AMD chips have a critical microcode bug that is about as bad as you can get. Microsoft decides not to fix a simple Windows bug that can cause a Blue Screen of Death or worse. And then we'll talk about certificate revocation. Turns out it's a hard computer science problem. And for some reason Let's Encrypt has decided to do it a different way. Stay tuned. Security Now! is coming up next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 987, recorded August 13th, 2024: Rethinking Revocation.

It's time for Security Now!, the show where we cover your security, your privacy, your safety, your online habitation at all times with this man right here, Steve Gibson of GRC.com. Hi, Steve.

Steve Gibson: Hello, Leo. Great to be with you again for, now, wait for it, the last episode of our 19th year.

Leo: Holy camoly. So you're saying as of next week...

Steve: Which will be the 20th, we will be here.

Leo: ...we'll be in our 20th year.

Steve: Our birthday will have been the previous day. Monday, August 19th of 2005 was...

Leo: A day that will live in infamy.

Steve: Episode number one, Honey Monkeys. Or was that number two?

Leo: Wow.

Steve: And, oh, I think that was all of 15-minute podcast. Oh, yes. Fortunately we don't have...

Leo: They've gotten longer ever since, unfortunately.

Steve: We don't have four ads in a 15-minute podcast, or I'd be like, uh, Leo, wait, wait, let me get a word in. No.

Leo: If you're interested, this is the episode, it's still online, it's 18 minutes and 10 seconds, and it's Security Now! Episode 1, August 18th, it says.

Steve: Oh, wait a minute. Okay, two days, yes, two days ahead we will have been...

Leo: Look at that. Do you want to hear how young we sounded back in the day? First of all, we had the worst music. Can you hear that? I can hear it, but I don't - oh, it's coming out of the wrong hole. Sorry about that.

Steve: That's what happens when you get older, Leo. That's right.

Leo: How many times have I heard that? Well, anyway, we'll figure that out. Actually, we'll leave it as an exercise for the viewer.

Steve: Perfect. It is there, if anyone's really dying to know what it was like 20 years ago, or 19. Wow.

Leo: It was us kind of staring into the headlights, actually.

Steve: Yeah, what are we going to do now? What is this, a what cast?

Leo: What the hell is this?

Steve: When you first said this, "How would you like to do a weekly podcast?" I said, "A what cast?"

Leo: Well, if you want to go directly there, TWiT.tv/sn1 is for now and forever the first Security Now!.

Steve: Yes. We'll see how long you leave this online, Leo. That remains to be seen.

Leo: Oh, you mean like after we retire and all that?

Steve: Yeah, exactly, yeah.

Leo: It is an interesting question that people talk about all the time is like, what happens to your digital life when you pass away? I don't know.

Steve: Wikipedia pages, they stay around. They say, "Yeah, we remember him."

Leo: Yeah. And, you know, a lot of what we do is on Archive.org. So maybe those live on in Archive.org.

Steve: Well, today, Episode, and I like the digits, 987. We're never going to get to 9876. I think that's asking too much. But 987, that we got. And no one is worried about 999 because we know that we're going to - I will have to spend a little time fixing my software to deal with four digits. That was always true. But, you know, I've got time. I'll make time.

Okay. So a bunch of stuff, fun stuff to talk about. It turns out that a million domains, it's estimated, are vulnerable to something known as the "Sitting Duck" attack. What is it?

Leo: That doesn't sound good.

Steve: It is not good. You do not want to be a sitting duck.

Leo: Oh, boy.

Steve: Is it new? Why does it happen? And who needs to worry about it? Also where the name came from is kind of fun. We'll get to that. Also we've got, believe it or not, another 9.8 CVSS remote code execution vulnerability discovered in Windows Remote Desktop Licensing service. Unfortunately, some 170,000 of those are exposed publicly. The good news is it was patched last month, that is, last month's Patch Tuesday. Today is August's Patch Tuesday. So if you happen to be a corporation, an enterprise of any kind who's been delaying updates, and you have Remote Desktop Licensing service publicly exposed, at least patch that. You know, you can go get the incremental patches from Microsoft if you want. That one, either turn off the service or patch it. Anyway, we'll talk more about that. I realize I forgot that I was doing a summary here.

Leo: Yes, this is just the short part, yeah.

Steve: Yes. Also, all of AMD's chips have a critical, but probably patchable, if they choose to do it, and apparently they're saying, eh, we're not really sure we're going to bother, microcode bug that allows boot-time security to be compromised. Okay. Also, Microsoft apparently decides NOT to fix a simple Windows bug which will allow anyone to easily crash Windows with a Blue Screen of Death anytime they want. Anyway, you don't want that in your startup folder. GRC's IsBootSecure freeware is updated. It's now a GUI, a regular Windows app, almost finished. I'll mention that briefly. And believe it or not, today's podcast 987 is titled "Rethinking Revocation"...

Leo: Oh, no.

Steve: ...because the entire certificate revocation system that the industry has just spent the past 10 years getting to work, is about to be scrapped in favor of what never worked before. Go figure.

Leo: Well, at least it never worked before.

Steve: Yeah, I think maybe we're just never happy with what we have, and it's like, well, okay, even though that didn't work before, maybe we can make it work now because what we have today has problems that we don't seem able to fix. Anyway, we're going to talk about that because everything's about to change. And of course - once again. And we've got a great Picture of the Week which kind of makes you wonder. So another great podcast for our listeners.

Leo: It's going to be a good one, kids. Stay tuned. Security Now!. I don't think there'll be much difference to this show because I'm in the attic, but Steve's in the same exact place. The only difference is we both have blinking lights now over our shoulders. Now let's get back to Steve and our Picture of the Week. I'm looking at it, and I don't - oh. Oh.

Steve: So, you know, like, the welcome mat that you have in front of your front door?

Leo: Yeah, yeah.

Steve: It says "Welcome" on it; right?

Leo: Yeah.

Steve: But to do that, you've got to be able to print the word. Now, what we have here is a - looks like a front door mat. And I've seen these before. You might have them in like an area that gets a lot of snow, where you'd like to be able to scrape your feet off. So this mat is a grid of holes which, you know, could be useful for scraping stuff off the bottom of your shoes.

Leo: Sure.

Steve: Now, the problem, however, for this particular instance is that someone thought that, you know, they'd hide a key under the mat.

Leo: No.

Steve: But, you know, the operative word in that phrase "hiding the key under the mat" is "hiding." And this mat, which is all holes, is a really bad hiding place for a key.

Leo: Yes. It's pretty obvious there's a key there, yeah.

Steve: It's really obvious, yeah.

Leo: That's great.

Steve: So anyway.

Leo: I hid the key under the mat. Well, I didn't hide it, but it is under the mat, yeah.

Steve: Another one of those weird "What were they thinking?" pictures that we entertain ourselves with every week here on this crazy podcast.

Okay. Every few years a security researcher rediscovers that many sites' DNS services are not properly configured and authenticated. To no one's surprise, until it happens to them, this leads to those misconfigured domains being taken over, meaning that the IP addresses being resolved from a domain name are pointed to malicious servers elsewhere. People going to those sites are often none the wiser. The domain name is right, and it looks like it's the right site. Little do they know that they're talking actually to a server in Russia.

So this generally doesn't happen to major sites because that problem would be seen and resolved almost immediately. But many large organizations register every typo variation of their primary domain name that they can think of in order to keep the so-called "typo squatters" from registering those almost-typed-correctly domains and thus catching people who mistype the actual domain name. But that's just an example of where you would have, like, lots of potential domains where, if it was being hacked, if something had been broken in the DNS, it wouldn't immediately come to any attention, but you could still get up to some mischief with it.

So the most recent round - as I said, every few years someone rediscovers this. The most recent round of this DNS hijacking surfaced recently when two security firms, Eclypsium and Infoblox, co-published their discovery. Now, as I said, though, it really should be considered a rediscovery - which is not to take anything away from them, we know that independent invention happens - since this issue does deserve as much attention as it can get, if for no other reason than as an industry we're still obviously having trouble getting this right. So, great, let's run the headlines again.

Eclypsium's write-up said: "Multiple threat actors have been exploiting this attack vector which we are calling Sitting Ducks since at least 2019" - and actually 2016, as we'll see in a second. But they wrote: "...2019 to perform malware delivery, phishing, brand impersonation, and data exfiltration. As of the time of writing" - which I think was yesterday - they wrote, "numerous DNS providers enable this through weak or nonexistent verification of domain ownership for a given account." Well, that can't be good. "There are an estimated one million exploitable domains, and we have confirmed more than 30,000" - and I did see 35,000 elsewhere - "hijacked domains since 2019. Researchers at Infoblox and Eclypsium, who discovered this issue" - okay, rediscovered - "have been coordinating with law enforcement and national CERTs since discovery in June 2024."

Okay, now, as I said, I'm glad this resurfaced. But we talked about this same issue eight years ago when on December 5th of 2016 security researcher Matthew Bryant posted his discovery under the title "The Orphaned Internet - Taking Over 120,000 Domains via a DNS Vulnerability in AWS, Google Cloud, Rackspace, and Digital Ocean." Now, I should just clarify, those weren't actually taken over. They were susceptible to being taken over. And that posting was Matthew's second of two postings about this problem.

So in this second of two postings, on December 5th, 2016, Matthew said: "Recently I found that Digital Ocean suffered" - you know, a major cloud hosting provider early on - "suffered from a security vulnerability in their domain import system which allowed for the takeover of 20,000 domain names. If you haven't given that post a read" - meaning his first one - he said, "I recommend doing so before going through this write up. Originally I had assumed that this issue was specific to Digital Ocean; but as I've now learned, this could not be further from the truth. It turns out this vulnerability affects just about every popular managed DNS provider on the web. If you run a managed DNS service, it likely affects you, too.

"This vulnerability is present when a managed DNS provider allows someone to add a domain to their account without any" - like a bad guy adding a domain to their bad guy account - "without any verification of ownership of the domain name itself." Which, again, why would anybody do that, but okay. "This is actually an incredibly common flow and is used in cloud services such as AWS, Google Cloud, Rackspace, and of course Digital Ocean. The issue occurs when a domain name is used with one of these cloud services, and the zone is later deleted without also changing the domain's nameservers. This means that the domain is still fully set up for use in the cloud service, but has no account with a zone file" - a set of DNS records is called a "zone file" - "to control it. In many cloud providers, this means that anyone can create a DNS zone" - meaning a set of DNS records - "for that domain and take full control over the domain."

Now, I'll just note that this sounds like that recent CNAME vulnerability we talked about where Microsoft lost control of a CNAME record because someone, you know, they deleted their service from Azure, but left DNS pointing to it. Yes, it's very much the same. And so what we're seeing is we're seeing this rush to everything being done in the cloud having some downstream consequences because the system really wasn't built for that. But why would that stop us?

So Matthew says: "This allows an attacker to take full control over the domain to set up a website, issue SSL and TLS certificates, host email, et cetera. Worst yet, after combining the results from the various providers affected by this problem, over 120,000 domains were vulnerable." And he says: "(Likely many more.)"

Okay. So in the recent reporting, it turns out that Russian hackers have been doing this for years, that is, actually doing this, sort of quietly behind the scenes, just getting up to what they get up to. This time around, Eclypsium and Infoblox have brought this to everyone's attention again, and giving it a catchy name, you know, Sitting Duck. Okay. Now, I particularly like this name since it nicely and accurately captures the essence, which is that a surprising number of domains are, in fact, as we say, sitting ducks. The naming was a bit of a stretch, but it's fun. The initials DNS, okay, Domain Name System, are repurposed to Ducks Now Sitting. Right.

Leo: Okay.

Steve: Thus Sitting Ducks. Okay.

Leo: Okay.

Steve: Ducks Now Sitting; that's right. That's what DNS stands for, kids. The trouble lies in the fact that responsibility is being irresponsibly delegated for something that's potentially as mission critical as one's DNS. Eight years ago, as I said, in Matthew Bryant's original posting, he explained the entire problem and provided means for its detection. He also responsibly disclosed his discovery to a bunch of cloud providers, those he enumerated, where, due to the nature of delegating DNS to them, the trouble is or was rampant.

Google Cloud DNS at the time had around 2,500 domains affected. They did what they could. Amazon Web Services, their Route 53 DNS - you know, they named it Route 53 because that's port 53 that DNS runs on - had around 54,000 domains affected so they performed multiple mitigations for the problem after this was brought to their attention. Digital Ocean had around, you know, those 20,000 domains we talked about earlier at risk, and they worked to fix what they could. And, interestingly, Rackspace felt differently than the others about the problem. They had around 44,000 domains affected, and they said this was not their problem.

Matthew explains how simple it is to take over an abandoned Rackspace domain. He wrote: "Rackspace offers a Cloud DNS service which is included free with every Rackspace account." Oh, well, then let's use the free one. Right. "Unlike Google Cloud and AWS Route 53, there are only two nameservers (dns1.stabletransit.com and dns2.stabletransit.com) for their cloud DNS offering so no complicated zone creation or deletion process is needed," he writes. All that needs to be done is to enumerate the vulnerable domains and add them to your account.

"The steps for this are the following: First, under the Cloud DNS panel, click the Create Domain button and specify the vulnerable domain and a contact email and Time To Live for the domain records. Step two, simply create whatever DNS records you'd like for the taken-over domain." And you're done.

Leo: What.

Steve: Yeah, it's like, ouch.

Leo: Oh, it's easy.

Steve: Yeah.

Leo: Press the Easy button on that one.

Steve: That's right. "This can be done," he writes, "for any of the 44,000 domain names to take them over. Rackspace does not appear to be interested in patching this (see below)," he writes, "so if you are a Rackspace customer, please ensure you're properly removing Rackspace's nameservers from your domain" - meaning your domain registration, wherever that be - "after you move elsewhere." I'll explain all that in a second.

So Matthew's blog posting provides the entire timeline of his interactions with the various providers. In the case of Rackspace, he initially notified them of this problem on September 9th, 2016. They responded on the 12th. So that was good. There was some back and forth for a few months until finally, on December 5th, after notifying Rackspace that he would be disclosing in 30 days, Rackspace replied: "Thank you again for your work to raise public awareness of this DNS issue. We've been aware of the issue for quite a while. It can affect customers who don't take standard security precautions when they migrate their domains, whether those customers are hosted at Rackspace or at other cloud providers.

"If a customer intends to migrate a domain to Rackspace or any other cloud provider, he or she needs to accomplish that migration before pointing a domain registrar at that provider's name servers. Otherwise, any other customer of that provider who has malicious intentions could conceivably add that domain to his or her account. That could allow the bad actor to effectively impersonate the domain owner, with power to set up a website and receive email for the hijacked domain. We appreciate your raising public awareness of this issue and are always glad to work with you." Have a nice day.

Leo: So we're not going to do anything about it.

Steve: Yeah, exactly. But in the meantime, you know, good luck to you. So in other words, yes, it's entirely possible for someone to set up their DNS incorrectly. And if they do so, others who share the same DNS nameservers, such as generally happens within cloud providers, could register and take over their domain. Have a nice day. So, now, as I said, if this seems vaguely familiar, it's probably because we recently talked about that somewhat similar DNS record problem, you know, with Microsoft and Azure and that CNAME record pointing somewhere. During today's round of rediscovery of this existing problem, which has been exacerbated by the move to the cloud, Brian Krebs interviewed a guy named Steve Job. You know, he's still alive, and there's no "s" on his name, so just Steve Job. He is the founder and senior vice president of the firm DNSMadeEasy, and the interview was conducted by email.

Steve Job said, interviewed by Brian Krebs, that the problem isn't really his company's to solve - and of course, you know, he understands about it, too - noting that DNS providers who are also not the domain's registrars have no real way of validating whether a given customer legitimately owns the domain being claimed. He wrote: "We do shut down abusive accounts when we find them, but it's my belief that the onus needs to be on the domain registrants themselves. If you're going to buy something and point it somewhere you have no control over, we can't prevent that."

So, okay, here's the bottom line takeaway from all of this. Despite all the press it has received, largely because they gave it a great name this time, you know, Matthew didn't name it. But now we've got Sitting Ducks. So, okay. We've got something we can hang onto. The problem is not new. The registration of a domain name inherently contains pointers to the domain's DNS nameservers that are providing the records for that domain. My registrar, Hover, manages the domain records for GRC.com and points to GRC's nameservers since I run my own DNS. Anyone who does that is safe from any of these problems. The other way to be safe is to have one's domain registrar, like Hover, also provide the domain's DNS servers and services. That's the optimal solution for most people. That has the advantage of keeping both the registration pointers and the servers to which they are pointing all in the family.

The danger that Matthew discovered and first raised back in 2016, and which Eclypsium and Infoblox have highlighted again, occurs when a domain's registrars are pointing to DNS servers that are being shared among tens of thousands of others. If an account in that shared cloud environment is discontinued while the external domain's registration is still pointing to the cloud's nameservers, anyone else can register that domain in that cloud and start receiving all of its traffic.

So as I noted before, this is unlikely to be happening with anyone's primary accounts since those accounts would tend to be well watched. But DNS for low-priority domains such as hundreds of excess typo-squatting prevention accounts which might have a server in the cloud which is just being used to bounce incoming traffic over to the correct domain, they could be, dare I say, sitting ducks. So just something to be aware of. If you're registered somewhere with your DNS, yet your servers are elsewhere, you want to be very sure with how you handle repointing your registration, the idea being the only safe way to do it is to set up DNS where you're moving to first, and then change your registration records to point to the new DNS. At no time do you want your registration name servers pointed somewhere where you do not have DNS, pointed to a cloud service.

> **Leo:** So I'm confused. Because I do this all the time. So when they say "migration," I don't understand what that means. So I have my DNS, my domain name company is Hover, right, like yours.

Steve: Yup.

> **Leo:** But I often want the DNS to be hosted by Fastmail. So what I do is I go in the DNS record at Hover, and I make sure that it says in the DNS servers that it's Fastmail servers.

Steve: So you have a MX record in Hover for...

> **Leo:** It's everything. So Hover, you can go into the full DNS, or you can just say, well, who's your DNS provider, and you say my DNS provider is Fastmail. At which point none of the other DNS settings apply. Fastmail has CNAME, has MX, it has A, everything. Right?

Steve: Right.

Leo: So where's the risk, is what I'm worried about. Because I do that all the time.

Steve: So what that would mean is that your Hover - so a domain registration has, like, the root of it is two name servers.

Leo: Right.

Steve: Which are something. And presumably it would be something dot Fastmail dot com.

Leo: Yeah, ns1.fastmail.com, ns2.fastmail.com.

Steve: Right. Okay. So the danger would be that your account at Fastmail would ever disappear.

Leo: Oh, oh, I see.

Steve: Or you would move it somewhere else.

Leo: Right.

Steve: And you would do that...

Leo: But without repointing it.

Steve: Yes, exactly.

Leo: I see.

Steve: So your registration would still be pointing to Fastmail, even though you no longer had DNS services from Fastmail.

Leo: Right.

Steve: And that would allow someone else to swoop in and register an account, saying, you know, this is my domain.

Leo: Ah. Because there's no authentication...

Steve: Right.

Leo: ...when I'm pointing over to Fastmail.

Steve: Right.

Leo: Ah.

Steve: Right. Fastmail relies on you to point to them.

Leo: I see.

Steve: They're not responsible for verifying that you, you know, the ownership of the domain that they're hosting. They assume that...

Leo: You must own it. You pointed it.

Steve: They assume your ownership because you were able to point it to them.

Leo: Right.

Steve: Exactly.

Leo: So if you point it and forget, then you're in trouble. And move your account out. Okay.

Steve: Then you would be a sitting duck.

Leo: Well, and I do this with Cloudflare, as well, because, again, Hover's the registrar, but sometimes I want things hosted on Cloudflare.

Steve: Yup.

Leo: So I will point the servers - now, the nice thing about Cloudflare, I'm sure they do this right. Fastmail really only has two servers, ns1 and ns2. Cloudflare it's a different server every time that you point to. They have many, many DNS servers.

Steve: Right.

Leo: So does that help? It helps.

Steve: It makes taking it over, well, it's a good question because Matthew specifically said that Rackspace only had two specific nameservers. So that means...

Leo: Right. That's what raised my interest because that's the same as Fastmail.

Steve: Yeah. But however Cloudflare has things set up, normally your domain registration is just a pair of nameservers. I don't know if I...

Leo: It is, but it's a different - but it's like a serialized - it's like a GUID.

Steve: Ah. Then that would probably be good because that suggests that...

Leo: It's unguessable.

Steve: ...that they're not all clustered under one destination server somewhere.

Leo: That makes sense.

Steve: So it sounds like it has your account name, you know, built into the nameserver name.

Leo: I would expect Cloudflare to do it right.

Steve: Yeah.

Leo: Yeah.

Steve: Speaking of doing it right, Leo, we're 30 minutes in. Let's take a pause. And then, believe it or not, we have another 9.8 remote code execution vulnerability courtesy of Microsoft.

Leo: Oh, no.

Steve: Oh, yes.

Leo: Oh, no. All right. Thank you, Steve.

Steve: They did patch it, but you've got to update.

Leo: Yeah.

Steve: Did I mention that this zero-day exploit has a proof of concept released publicly? I may have missed that.

Leo: That makes it really dangerous; right? Because any script kiddie now...

Steve: Yeah. Microsoft, yeah, when you're a Microsoft shop, headlines which read "Exploitable proof of concept released for a zero-click remote code execution..."

Leo: Oh, it's zero-click. You didn't mention that, either. Oh, boy.

Steve: Zero-click. Yes, because we wouldn't want to put anyone out by asking them to click their mouse to be taken over, no.

Leo: Holy cow.

Steve: Yeah, "...zero-click RCE that threatens all Windows servers." Now, okay, my heart skipped a beat because I am a Microsoft shop.

Leo: Yeah.

Steve: And it's like from Windows, like, 2000 all the way through 2025, all of them. So if your organization is publicly exposing the Windows Remote Desktop Licensing Service (RDL) - and the good news is I'm not exposing anything, so then I went <sigh>, okay, I'm okay - then it's hopefully not too late for you to do something about it. Three security researchers have detailed and published proof-of-concept exploit code for a critical CVSS 9.8 vulnerability being tracked as CVE-2024-38077. It's referred to as "MadLicense."

This 9.8er impacts all iterations of Windows Server from 2000 through 2025. And it's not even 2025 yet, it's 2024. In other words, all of them. And if your service is publicly exposed, that's as bad as it gets. It's a pre-authentication remote code execution vulnerability that allows attackers to seize complete control of a targeted server without any form of user interaction. So it's remote and zero-click and affects all Windows services which are exposing the Windows Remote Desktop Licensing Service.

This RDL service is responsible for managing licenses for Remote Desktop Services, as its name sounds, and it's consequently deployed across many organizations. Security researchers have identified a minimum of 170,000 RDL services directly exposed to the Internet, which renders them all readily susceptible to exploitation. And, you know, at some point I'll get tired of saying that it just is not safe to expose any Microsoft service except web, you know, because there you have no choice. But put it behind some sort of protection, a VPN, some extra filtering of some sort. You know, port knocking, anything.

But don't just have a Microsoft service on the public Internet. One after the other they are found to be vulnerable, and people are being hurt. Anyway, the good news is, well, actually it's not the good news. But in terms of limiting the total number, 170,000, not good, but the nature of this is that it's going to be larger enterprises which are exposed. And unfortunately, those are the tasty targets.

Okay. So this so-called "MadLicense" vulnerability arises from a simple heap overflow. By manipulating user-controlled input, attackers can trigger a buffer overflow which leads to arbitrary code execution within the context of the RDL service. Researchers have successfully demonstrated a proof-of-concept exploit on Windows Server 2025, achieving a near 100% takeover success rate. The exploit effectively circumvents all contemporary mitigations. There were some details that I didn't want to drag everyone through, where the latest Server 2025 has, you know, so-called advanced protection mitigations. This just ignores them.

So while the proof of concept demonstrated the vulnerabilities exploitation on Windows Server 2025, the researchers emphasized that the bug could be exploited more quickly and efficiently on older versions of Windows Server. And of course there's lots of those, and those have fewer mitigations in place. The proof of concept was designed to load a remote DLL. Wow. But the researchers noted that, with slight modifications, it could execute arbitrary shellcode provided by the attacker within the RDL process, making the attack even stealthier, meaning that the infected system would not be reaching out onto the Internet to obtain and download that malicious DLL, which might give it away. So just provide your own shellcode to do what you need.

The researchers did responsibly disclose the vulnerability to Microsoft a month before going public. And while Microsoft patched the flaw in last month's security Patch Tuesday, meaning July's

Patch Tuesday, the fact that it was initially marked, lord knows why, as "exploitation less likely" highlights the potential for underestimating such threats. You know, Microsoft would like to say, oh, don't worry about that. But, you know, wow. They should be protecting their customers. Doesn't seem like they are in this instance.

So anyone who's not actively using and depending upon this Remote Desktop Licensing Service, who does have it publicly exposed, should definitely shut it down immediately. And if for any reason you're a month behind on updates, since as I mentioned at the top of the show today is August's Patch Tuesday, so now there's two months' worth of patches, both which fix this, you should seriously consider at least applying the patch for this one. There is a proof of concept posted on GitHub, so we know it's not going to be long before the bad guys are attacking any unpatched network. And we know there will be some, for reasons which defy belief or understanding.

Okay. This next bit of news would have been today's main topic if I didn't think it was also important and very interesting to catch up on what's been happening in the important and still troubled world of certificate revocation. We opened this door last week, that is, the revocation door last week after DigiCert's mass revocation event and my own "revoked.grc.com" server event, which we'll catch up on. It turns out that in researching this further I've found that everything we've been doing for certificate revocation over the past 10 years is about to change again. Okay. So we'll get to that.

For now, let's talk about the big runner-up this week, which was dubbed "Sinkclose." It was dubbed Sinkclose by the two security researchers who discovered it. We've just been talking about Secure Boot and how the AMI BIOS's never-meant-to-be-shipped sample Platform Key was indeed shipped in as many as 850 different makes and models of PCs.

Leo: You mean the one that said "Don't trust this key"?

Steve: Yes. DO NOT SHIP, DO NOT TRUST. And if you look at the certificate for it, it says that right there on the front page. It's like, okay, well, I guess no one looked at that. Another thing that's not good is when two researchers discover a fundamental vulnerability in AMD's processors - and by that I mean nearly all of AMD's processors - which allows for the subversion of this same Secure Boot process, regardless of what key its platform is using.

Leo: Well, that makes it easier.

Steve: And AMD is a popular processor, as we know.

Leo: Oh, yeah. I have one right over here.

Steve: Uh-huh. Okay. So WIRED covered this discovery and gave it the headline "'Sinkclose' Flaw in Hundreds of Millions of AMD Chips Allows Deep, Virtually Unfixable Infections." Now, that's a bit of an exaggeration. But, you know, in fairness, we seem to be encountering similar "sky is falling" headlines these days almost weekly. So what's going on? Their subhead was not any more encouraging. It read "Researchers warn that a bug in AMD's chips would allow attackers to root into some of the most privileged portions of a computer, and that it has persisted in the company's processors for decades."

Okay. So I'm going to first share a somewhat trimmed-down version of what WIRED wrote because we have a more sophisticated audience. But then we'll see where we are. So WIRED explained: "Security flaws in firmware have long been a target for hackers looking for a stealthy foothold. But only rarely does that kind of vulnerability appear not in the firmware of any particular computer maker, but in the chips found across hundreds of millions of PCs and servers.

Now, security researchers have found one such flaw that has persisted in AMD processors for decades, that would allow malware to burrow deep enough into a computer's memory that, in many cases, it may be easier to discard a machine than to disinfect it."

Leo: Oh, I'm not discarding my game machine. Hold on there, buddy.

Steve: "At the DEFCON hacker conference" - which is underway right now - "researchers from the security firm IOActive" - that we've spoken of many times in the past - "plan to present a vulnerability in AMD chips they're calling Sinkclose. The flaw allows hackers to run their own code in one of the most privileged modes of an AMD processor, known as System Management Mode." That's even below ring 0. That's, like, the code that has the lights on the motherboard even when nothing else is running. WIRED wrote: "...designed to be reserved only for a specific, protected portion of its firmware. IOActive's researchers warn that it affects virtually all AMD chips dating back to 2006, or possibly earlier. The researchers, Nissim and..." Uh. I've practiced pronouncing this before. But it's like, it's O-K-U-P-S-K-I.

Leo: Oh, sure, that's Okupski.

Steve: Okupski.

Leo: Okupski. Okupski.

Steve: Reminds me of Shabubi. But no.

Leo: The key in all of these is, if you say it in the Boris Badenov accent, it works.

Steve: You can get away with anything.

Leo: You can get away, too. Okupski. That's Okupski, yeah.

Steve: So the researchers, thank you, Nissim - or, no, wait, yeah, Nissim and Okupski, "note that exploiting the bug would require hackers to have already obtained relatively deep access to an AMD-based PC or server, but that the Sinkclose flaw would then allow them to plant their malicious code far deeper still. In fact, for any machine with one of the vulnerable AMD chips" - and again, everybody who has AMD chips are vulnerable, essentially - "the IOActive researchers warn that an attacker could infect the computer with malware known as a 'bootkit' that evades antivirus tools and is potentially invisible even to the operating system, while offering a hacker full access to tamper with the machine and surveil its activity.

"For systems with certain faulty configurations in how a computer maker implemented Platform Secure Boot, which the researchers warn encompasses a large majority of the systems they tested, a malware infection installed via Sinkclose could be harder yet to detect or remediate, they say, surviving even a reinstallation of the operating system." Right. We've talked about this; right? It's down in the firmware that boots the system. It is a bootkit.

"Okupski said: 'Imagine nation-state hackers or whoever wants to persist in your system. Even if you wipe your drive clean, it's still going to be there. It's going to be nearly undetectable and nearly unpatchable.' Okupski said: 'Only opening a computer's case, physically connecting directly to a certain portion of its memory chips with a hardware-based programming tool known as an SPI Flash programmer, and then meticulously scouring the memory, would allow the malware to be removed.'" And of course you'd have to know what was expected there because normally no

motherboard provider is going to let you have what should be in that flash chip. "Nissim sums up that worst-case scenario in more practical terms: 'You basically have to throw your computer away.'

"In a statement shared with WIRED, AMD acknowledged IOActive's findings, thanked the researchers for their work" - while silently cursing them - "and noted" - no, I added that - "and noted that it has 'released mitigation options for its AMD EPYC datacenter products and AMD Ryzen PC products, with mitigations for AMD embedded products coming soon." And WIRED said: "(The term 'embedded' in this case refers to AMD chips found in systems such as industrial devices and autos.) For its EPYC processors designed for use in data-center servers specifically, the company noted that it released patches earlier this year," because the researchers gave them plenty of time. "AMD declined to answer questions in advance about how it intends to fix the Sinkclose vulnerability, or for exactly which devices and when, but it pointed to a full list of affected products that can be found on its website's security bulletin page." And I do have a link to that below this in the show notes.

"In a background statement to WIRED, AMD emphasized the difficulty of exploiting Sinkclose." That is, you know, they're trying to keep everyone from hyperventilating. They said: "To take advantage of the vulnerability, a hacker has to already possess access to a computer's kernel, the core of its operating system. AMD compares the Sinkhole technique to a method for accessing a bank's safe-deposit boxes after already bypassing its alarms, the guards, and the vault door." On the other hand, "Ocean's Eleven." "Nissim and Okupski respond that while exploiting Sinkclose does require kernel-level access" - that's true - "to a machine, such vulnerabilities are exposed in Windows and Linux practically every month."

Leo: Yeah.

Steve: And we know that's true. "They argue that sophisticated state-sponsored hackers of the kind who might take advantage of Sinkclose likely already possess techniques for exploiting those vulnerabilities, known or unknown. Nissim said: 'People have kernel exploits right now for all these systems. They exist, and they're available for attackers. This is the next step.'" In other words, this is what they've been waiting for.

"Nissim and Okupski's Sinkclose technique works by exploiting an obscure feature of AMD chips known as TClose. The Sinkclose name comes from combining the TClose term with Sinkhole, the name of an earlier System Management Mode exploit found in Intel chips back in 2015. In AMD-based machines, a safeguard known as TSeg prevents the computer's operating systems from writing to a protected part of memory meant to be reserved for System Management Mode known as System Management Random Access Memory, or SMRAM. AMD's TClose feature, however, is designed to allow computers to remain compatible with older devices that use the same memory addresses as SMRAM, remapping other memory to those SMRAM addresses when it's enabled." In other words, and when you do that, you're no longer able to access the underlying memory because you've mapped other memory on top of it, and so that's what you now access. But there's a problem with that.

"Nissim and Okupski found that, with only the operating system's level of privileges, they could use that TClose remapping feature to trick the System Management Mode code into fetching data they've tampered with, in a way that allows them to redirect the processor and cause it to execute their own code at the same highly privileged System Management Mode level. Okupski said: 'I think it's the most complex bug I've ever exploited.'" And speaking of complex, I've avoided his first name because it doesn't have any vowels, so I can't even begin to pronounce it.

"Nissim and Okupski, both of whom specialize in the security of low-level code like processor firmware, said they first decided to investigate AMD's architecture two years ago, simply because they felt it had not gotten enough scrutiny compared to Intel" - which of course we've over-scrutinized, if anything - "even as its market share has risen. They found the critical TClose edge case that enabled Sinkclose, they said" - get this - "just by reading and rereading AMD's documentation. Nissim said: 'I think I read the page where the vulnerability was about a thousand

times. And then on the 1,001 I noticed it.' They alerted AMD to the flaw back in October of last year, but have waited nearly 10 months to give AMD ample time to prepare a fix" because they need to change the microcode in all of the affected AMD chips in order to fix this flaw in TClose.

"For users seeking to protect themselves, Nissim and Okupski say that for Windows machines, likely the vast majority of affected systems, they expect patches for Sinkclose to be integrated into updates shared by computer makers with Microsoft, who will roll them into future operating system updates. Patches for servers, embedded systems, and Linux machines may be more piecemeal and manual; for Linux machines, it will depend in part on the distribution of Linux a computer has installed.

"Nissim and Okupski say they agreed with AMD not to publish any proof-of-concept code for their Sinkclose exploit for several additional months to come" - now that this has everyone's attention - "in order to provide more time for the problem to be fixed. But they argue that, despite any attempt by AMD or others to downplay Sinkclose as too difficult to exploit" - well, everyone listening to this podcast knows better than that - "it should not prevent users from patching as soon as possible. Sophisticated hackers may already have discovered their technique." Right? I mean, we don't know that somebody else didn't study those pages and go, uh, wow, there's a problem here. "Or they could figure out how, now that they know there's a problem there, exactly how Nissim and Okupski present their findings once this has been presented at DEFCON." Again, it's now public.

"Even if Sinkclose requires relatively deep access, the IOActive researchers warn, the far deeper level of control it offers means that potential targets should not wait to implement any fix available. Nissim finished: 'If the foundation is broken, then the security for the whole system is broken.'" So as I said, I've got a link in the show notes. It's, like, all the chips that AMD makes. As we know, when we've covered this relative to the need for Intel to patch their microcode, Windows is able to, and fortunately does, patch the underlying processor microcode at boot time on the fly. It's somewhat annoying that there isn't anything more definitive from AMD about what's patched and what isn't, and when what isn't will be, since this really is a rather big and important vulnerability. It allows anything that's able to get into the kernel to drill itself right down into the firmware and live there permanently, period.

But, you know, we may need to wait a few months until the IOActive guys are able to disclose more. And hopefully they'll produce a definitive test for the presence of the patch, or someone will create a benign test for the vulnerability, which would be very useful. So, wow, another bad...

> **Leo:** That's something. Jeez.

Steve: Bad biggie. And Leo? We're at one hour. Let's take a break. And then we're going to look at, okay, the most bizarre thing imaginable, a user - a trivial to reproduce Blue Screen of Death allowing anyone who wants to, to crash Windows. Which Microsoft said, well, it didn't happen to us, so what? What are you talking about? And they're not fixing it.

> **Leo:** Okay. Your mileage may vary, I guess. I'm glad you're here. We're glad you're watching Security Now! with Steve Gibson. We do it every Tuesday. It's kind of the day of the week that most security professionals wait for all week long to find out what's going on in the world of security. And this show is no exception. Wow. Holy cow. This one's wild.

Steve: This is. Okay. So we have yet another example, we've unfortunately had a few, of Microsoft apparently choosing, deliberately choosing, not to fix what appears to be a somewhat significant problem. This allows anyone to maliciously crash any recent and fully patched Windows system, and that would seem to be significant. Yet Microsoft doesn't appear to think so. Or they can't be bothered.

This all came to light just yesterday, and the Dark Reading site has a good take on it. They wrote: "A simple bug in the Common Log File System driver" - that's CLFS, so it's clfs.sys, the Common

Log File System driver - "can instantly trigger the infamous Blue Screen of Death across any recent versions of Windows. CLFS," they write, "is a user- and kernel-mode logging service" - which is why any user can do it - "that helps applications record and manage logs. It's also a popular target for hacking." And if you google clfs.sys, you'll see it's had a troubled history. So you'd think Microsoft would go, oops, again? Apparently not.

They write: "While experimenting with its driver last year, a Fortra researcher discovered an improper validation of specified quantities of input data which allowed him to trigger system crashes at will. His proof-of-concept exploit worked across all versions of Windows tested including 10, 11, and Windows Server 2022 even in the most up-to-date systems. The associate director of security R&D at Fortra said: 'It's very simple: Run a binary, call a function, and that function causes the system to crash.' He added: 'I probably shouldn't admit to this, but in dragging and dropping it from system to system today, I accidentally double-clicked it and crashed my server.'"

Leo: Oh, that sounds like something I'd do.

Steve: "The underlying issue" - which has been given a CVE-2024-6768 "concerns Base Log Files (BLFs), a type of CLFS file that contains metadata used for managing logs."

Leo: Ah, it's an interpreter.

Steve: The clfs.sys driver, it seems, does not adequately validate the size of data within a particular field in the BLF. Okay, so, right? So this is a metadata file which is used to control the CLFS. And it's assuming the metadata is correct because why wouldn't it be?

Leo: Why wouldn't it be.

Steve: Who would ever tamper with that?

Leo: Ever.

Steve: Right. Any attacker with access to a Windows system can craft a file with incorrect metadata size information which will confuse the driver. Then, unable to resolve the confusion, it triggers KeBugCheckEx, the function that triggers the Blue Screen of Death.

Leo: Closure.

Steve: This CVE has a, you know, it's a medium badness; right? It's a 6.8 out of 10 on the CVSS scale. It doesn't affect the integrity or confidentiality of data, nor cause any kind of unauthorized system control, although it probably could ground Delta Air Lines, when you think about it. It does, however, allow for wanton crashes that can disrupt business operations or potentially cause data loss.

"Or, as Fortra's associate director of security R&D explains: 'It could be paired with other exploits for greater effect. It's a good way for an attacker to maybe cover their tracks, or take down a service where they otherwise shouldn't be able to, and I think that's where the real risk comes in,' he says. 'These systems reboot unexpectedly. You ignore the crash because it came right back up and, oh, look, it's fine now. But that might have been somebody hiding their activity, hiding the fact that they wanted the system to reboot so that it could load some new settings or have them take effect.'

"Fortra first reported" - get this - "its findings last December 20th," five days before Christmas. Here's a present for you, Microsoft. "After months of back and forth," they say, Fortra said, "Microsoft closed their investigation without acknowledging it as a vulnerability or applying a fix. Thus, as of this writing yesterday, it persists in Windows systems no matter how updated they are.

"In recent weeks, Windows Defender has been identifying Fortra's proof of concept, which apparently they provided to Microsoft, as malware." Oh, so don't do that. Which does not fix the problem. "But besides running Windows Defender and trying to avoid running any binary that exploits it, there's nothing organizations can do with CVE-2024-6768 until Microsoft releases a patch. Dark Reading has reached out to Microsoft for its input, but has heard nothing back."

Okay. So get a load of this timeline, which Fortra published on their vulnerability disclosure report: December 20th, 2023: Reported to Microsoft with a proof-of-concept exploit. So, hey, Microsoft, I know it's close to Christmas, but here's some code that crashes everything. Um, just thought you should know. January 8th: Microsoft responded that their engineers could not reproduce the vulnerability. Wonder what version of Windows they're using? January 12th, four days later: Fortra provided a screenshot showing a version of Windows running the January Patch Tuesday updates - meaning, okay, we patched in January Here's the memory dump after it crashes.

February 21st: Microsoft replied that they still could not reproduce the issue, and they were closing the case. A week later, February 28th: Fortra reproduced the issue again with the February Patch Tuesday updates installed and provided additional evidence, including a video of the crash condition. Maybe they should just put it on YouTube. Okay. So that's February 28th.

Now we zoom, fast-forward to June 19th, 2024: Fortra followed up to say that, they said, "We intended to pursue a CVE and publish our research." A month, almost a month goes by. Now we're on July 16th: Fortra shared that it had reserved CVE-2024-6768 and would be publishing soon. August 8th: Reproduced on latest updates, that's the July Patch Tuesday, last month's Patch Tuesday, of Windows 11 and Server 2022 to produce screenshots to share with the media. August 12th, yesterday, is the CVE publication date.

So this is the same company, meaning Microsoft, that wants to continuously record everything every Windows PC user does while using Windows while assuring us that our entire recorded history will be perfectly safe. No, thank you. Oh, and I did see on GitHub that what was a Windows crash has now evolved with this flaw into an elevation of privilege.

Leo: See? Isn't that what happens? That's why you never can say, oh, it's just a crash.

Steve: Yup.

Leo: Because the next step after the crash is, well, now we've got - what do we do to control a system now.

Steve: Now we have an unprivileged user, any, apparently, unprivileged user is able to elevate themselves to full root.

Leo: And that's how it always progresses.

Steve: Yup.

Leo: That's why you take that stuff seriously.

Steve: Yup. Crazy. Okay. So as we know, two weeks ago we covered the news from Binarly that an estimated 10% of PCs had shipped with, and were still currently using and relying upon AMI's sample demonstration UEFI Platform Keys which had never been intended to be shipped into production, blah blah blah. Secure Boot isn't. Binarly named it "PKfail." The following Friday I saw that it was not easy for anyone to determine whether they had these keys. By last Tuesday I had a Windows console app finished and published, and I said I would have the graphical version this week. And I do. I'm going to, after the podcast today, there's a problem with foreign language installs of Windows. I know how to fix it. I just didn't have time before the podcast to do so.

So the first release is there. Works for everybody running English version of Windows. And it kind of works for everybody else in the world, but not the way I want it to. So there's a cosmetic problem that I'm going to fix. So IsBootSecure is available for download from GRC. You're able to run it. It tells you immediately what your situation is. Are you BIOS or UEFI? If you're UEFI, is Secure Boot enabled or not? And if it's enabled, it'll show you your cert. Many BIOSes will show you your cert even if Secure Boot is not enabled. Not that it matters, but still it's nice to know. Some BIOSes don't. So it tells you that either way.

And if it is able to get the cert, it'll show it to you and allow you to browse around and look at the information in it. And if for some reason you want to write it to a file, you're able to export the cert from your firmware using IsBootSecure, as well. So anyway, a cute little freebie, and case closed. We'll see how this evolves over time.

Leo: So is it like a real graphical interface? Or is it just a TUI? Is it just text...

Steve: No, no, it's a dialog. Yeah, in fact if you go to IsBootSecure.htm, you'll see two sample dialogs, and you can bring it up on our screen while we're talking about it. GRC.com/isbootsecure. And it ought to...

Leo: Do I have to have the htm, or...

Steve: No, you don't, because I put that on for you.

Leo: How does he do that? And here it is.

Steve: Yeah.

Leo: Yeah.

Steve: And so it shows you that it is or is not enabled.

Leo: Nice.

Steve: And whether or not you've got good keys or keys that say DO NOT TRUST/DO NOT SHIP. Yeah.

Leo: And it's a freebie.

Steve: Yeah. 33, is it 3,300 downloads so far?

Leo: Wow.

Steve: So, yeah, it's taking off nicely.

Leo: And it's only just begun.

Steve: Yeah. And what's really cool is that I did the mailing to Security Now! listeners this morning. 8,145 of our listeners, 8,145 received the email hours ago, and I've already had feedback from them. So even before the podcast is happening we're getting feedback on the podcast.

Leo: You've got a good group, I have to say.

Steve: Yeah, we really do.

Leo: It's really impressive. If you're not on the email list, go to GRC.com/email. And it's actually got two benefits. You can sign up for emails, but you can also validate your own address so that you can from then on email Steve, so if you have feedback or...

Steve: Yes. Then, exactly. And, boy, did I get feedback when what I'm about tell you happened, happened.

Leo: Oh. All right. Let's hear about this Revocation Revoked.

Steve: Oh. Last week we covered DigiCert's immediate revocation of more than 83,000 certificates after it was discovered that their automation software had been found to not be prefixing an underscore onto the front of randomly-generated 32-character subdomain names which were being used in DNS CNAME records to create proof of control over those domains. So, you know, you would say to DigiCert, hey, I want a certificate for Jimmy's Tire Works. And they'd say, okay, here's a blob, a random string. Add this to Jimmy's Tire Works' DNS. When you have done that, let us know. We will query that record with that random string. And if we can, then that means you actually have control of Jimmy's Tire Works' DNS, and we're going to give you a certificate. So that's proof of control over a domain.

At the end of that discussion last week we talked about the historical mess of certificate revocation, and I reminded everybody about GRC's "revoked.grc.com" test website which I originally created back in 2014, 10 years ago, when this podcast covered this topic in depth. And to bring, in order to talk about it, to bring the "revoked.grc.com" site back to life for last week's podcast, the previous week, on Thursday, August 1st, I had created and then immediately revoked a brand new certificate for the "revoked.grc.com" web server.

And as we all know, five days later on August 6th, one week ago today, during last week's podcast, Leo and I and all of our online listeners at the time were verifying that, sure enough, that site, which was serving a certificate that had been revoked five days earlier, which Ivan Ristic's SSL Labs site verified, Leo, you brought that up while we were on the air, verified was revoked. Despite that, that site was still being happily displayed by everyone's browsers of every ilk on every platform, even those that were set to use OCSP, the Online Certificate Status Protocol.

As luck would have it, the next day I began receiving email from our listeners informing that none of their various web browsers would any longer display the revoked.grc.com site.

Leo: Yeah, I just tried it on Arc on Mac.

Steve: What do you know.

Leo: And this is what you'd want to see; right? That's a bad certificate, buddy.

Steve: It is revoked. And it even says there net::err_cert_revoked.

Leo: Yeah.

Steve: So for everybody it was showing as an untrusted site with a revoked certificate. Okay. What happened?

Leo: Yeah, one day, overnight.

Steve: What happened is that, five days before, that is, five days before the podcast, when I was bringing this up, with DigiCert's help I created a valid certificate for the domain "revoked.grc.com" and installed it on its server. Then I surfed over to that site with my browser to verify that everything was good. Like, you know, the server was up; that, you know, the page came up and everything was fine.

In the process of displaying that page, behind the scenes the revoked.grc.com web server which is OCSP-stapling capable examined its own certificate and found DigiCert's Online Certificate Status Protocol server's URL which is at http://ocsp.digicert.com. And note that the URL actually is HTTP because the service that's behind this URL has no need for any protection from spoofing or privacy. It's http://ocsp.digicert.com.

Upon finding that URL, five days before the podcast, GRC's revoked server queried DigiCert's OCSP server for the current status of this brand new certificate. And because I was testing the server to make sure the certificate was installed and working properly, I had not yet revoked it. So DigiCert's OCSP response was to return a short-lived, seven-day, timestamped certificate attesting to the fact that this bright and shiny new certificate was valid as the day it was born.

Upon receiving that certificate, GRC's revoked server stored that brand new OCSP certificate in a local cache. And from then on, every time anyone's web browser attempted to connect to revoked.grc.com, GRC's server would not only send out its own revoked.grc.com certificate, but "stapled to" that certificate was DigiCert's OCSP certificate having a valid expiration date of August 8th, one week from the date it was first asked to produce an OCSP certificate for the "revoked.grc.com" site.

So having then seen that everything looked good and was working as I expected, I returned to DigiCert and immediately manually revoked the brand new certificate. I checked DigiCert's own site's status page for the "revoked.grc.com" domain, and it confirmed revocation.

I went over to Ivan Ristic's excellent SSLLabs.com facility and verified that it was very unhappy with the revoked.grc.com site. So everything looked fine. But because GRC's server, as are many web servers today, is OCSP stapling-capable, it was proudly stapling to its revoked certificate the original and not-yet-expired pre-revocation OCSP certificate it had received from DigiCert the first time it was asked. Whoops!

And just as today's web servers are now stapling OCSP certificates to their own certificates, today's web browsers are now relying upon those stapled OCSP certificate assertions since it means that they do not need to look any further. They don't need to bother making their own

independent queries to the certificate's authority to check on the current status of the certificate because there's a freshly stapled short-life certificate from that Certificate Authority saying yes, the certificate is good. Except when it's not.

As a consequence of that, last Tuesday, everyone's browser going to the revoked.grc.com website received not only the recently minted revoked.grc.com certificate that was valid for the next 13 months, but also a valid and unexpired OCSP assertion, "stapled" to that certificate, stating that it was, as of the date of the OCSP certificate, the revoked.grc.com certificate was valid and in good standing. So no one's normally configured web browser would, or did, look any further. Here was a recent, fresh, unexpired assertion, signed by DigiCert themselves, stating that all's good with the cert.

But on Wednesday, a day before its cached copy of that original OCSP certificate was due to expire, GRC's revoked server reached out to DigiCert for an update, to update its cached OCSP stapling. Web servers that do OCSP stapling typically begin asking for an OCSP status update well before their existing OCSP certificate is due to expire so that they're never caught flat-footed without one, and are certain to obtain a replacement from their Certificate Authority well before the current certificate expires. That happens a day before, typically.

And at that point, six days after the certificate's creation, last Wednesday, GRC's revoked server learned that the certificate it was serving had been revoked. So it got out its stapler and replaced the soon-to-expire original OCSP certificate with the new one, carrying the news to everyone who was asking from that point on.

So what we've all just witnessed is a perfect example of a pretty good compromise.

Leo: Yeah, it worked.

Steve: Well, it worked. And it wasn't immediate; but it was, you know, six days.

Leo: Within the week, yeah.

Steve: Yeah. Okay. So to see what I mean by that, let's turn the clock back 10 years to see how we got here. We first looked at all this 10 years ago, in 2014, on this podcast. At that time the concept of OCSP stapling existed, but it was still not widely deployed. I recall talking about it wistfully, since it really did seem like an ideal solution. But it still wasn't widely supported.

Our long-time listeners will recall that OCSP services themselves did exist 10 years ago, but their services were not used by default. As we know, again, OCSP stands for Online Certificate Status Protocol. And the idea behind it is elegantly simple: Any Certificate Authority that's issuing certificates runs servers at a public URL that's listed in every certificate they sign, so anyone can know how to find it. This allows anyone who receives a certificate that the CA signed to obtain the URL from within the certificate and query the Certificate Authority's OCSP service right then and there to obtain up-to-the-moment information about any specific domain's certificate.

And I'll just note that, if browsers were doing that today instead of relying on the stapled and potentially up to a week-old OCSP certificate, everyone's browser would have immediately complained, I mean, like immediately after I revoked the site. But browsers aren't doing that now. They're relying on stapling when it exists. So what the OCSP service returns is a timestamped, short-lived, signed-by-the-Certificate Authority attestation of the current status of the queried certificate. It created, as its name says, an Online Certificate Status Protocol.

At the time, 10 years ago, we played with Firefox's settings, which could cause Firefox to query Certificate Authorities' OCSP servers in real time. But there were a couple of problems with this. One problem was that OCSP services back then were still not very reliable and could be slow to respond. So a browser visits a website, like the main website the visitor wants to go to; receives the site's TLS, back then SSL, certificate; examines that certificate to obtain its issuing Certificate

Authority's OCSP server URL; and queries the URL for a real-time verification of the certificate's validity.

Even if the OCSP server responds quickly, a cautious web browser should wait before doing anything with the website's page since the presumption is that the certificate it was served might be bogus. This is, after all, the reason for the OCSP double-check is to make sure. So this could introduce a significant delay in the user's experience that no one wants. And what if no one is home at the OCSP service? What if the CA is having a brief outage, or they're rebooting their service, or updating it, or who knows what? Does the web browser just refuse to connect to a site that it cannot positively verify?

The goal, of course, is to never trust an untrustworthy site. But given the fact that nearly all checked certificates will be good, and that only the very rare certificate will have been revoked before it expires on its own, it seems wrong to refuse to connect to any site whose Certificate Authority's OCSP service doesn't respond or might be slow. As a consequence, 10 years ago, even when a web browser had been configured to check OCSP services, unless it was forced to do otherwise, browsers would generally fail open, meaning they would default to trusting any site that they were unable to determine should definitely not be trusted.

So one clear problem was the real-time trouble that real-time certificate verification created. Recall that back then Google was, like, on a tear about, like they were frantic about the performance of their web browser, Chrome, which was still kind of new. The last thing they were going to do was anything that might slow it down in any way. So they created these bogus CRL sets, and I called them out on it because they were saying, oh, yeah, this works. And I said, "No, it doesn't."

Leo: Yeah, we talked about this.

Steve: Oh, yeah. Believe me, I...

Leo: Yeah.

Steve: It was a number of podcasts, and it came back a few times. Okay. But another problem was the privacy implications which this brought to the web browser's user. If every site someone visits causes their browser to reach out to remote Certificate Authority-operated OCSP servers, then anyone monitoring OCSP traffic - which as I mentioned has always been unencrypted HTTP, and even is today - that would allow anyone monitoring OCSP traffic to track the sites users were visiting...

Leo: [Crosstalk]

Steve: ...since their browsers, yeah, their browsers would be continually querying for every site's current certificate status. This was a huge concern for the privacy of the Internet's web users.

So the reason I was so excited about OCSP stapling 10 years ago was that it beautifully addressed and resolved all of these problems. As you've probably figured out from what I've already said, OCSP stapling flips the original OCSP model on its head. With stapling, instead of having the user's web browser going out to continually fetch every site's current OCSP status, the same site that's offering its certificate "staples" a recent and unexpired OCSP response from the Certificate Authority, just as a browser would retrieve it. Instead, the site staples that to its own certificate.

Now the web browser doesn't need to perform another fetch and query. So there's no delay, no extra fetching and waiting, and no privacy issue because, you know, the browser's already going to the site and might as well get confirmation that the certificate's good. Essentially the server is saying "Here is my certificate, and the Certificate Authority that issued it no more than a week

before has asserted that as of that time it's good."

What we learned over the past week is that the phrase "much more recent," that is, the CA's much more recent assertion might not always be as recent as we would hope. That is to say, it's not immediate. The OCSP response for every domain I have checked - mine, Google's, Let's Encrypt, Hover's, Facebook's (actually Facebook is also using DigiCert, so GRC uses the same OCSP services as Facebook) - they are all, all of those Certificate Authorities are issuing OCSP response certificates with a seven-day life.

And this is why I referred to this earlier as a pretty good compromise. The reason we need some compromise is that for practical purposes we need to tolerate some local caching of previously received and unexpired OCSP responses. Ten years ago, the entire world had not yet switched over to using HTTPS all the time for everything. Now it has. Today, virtually every of the many connections that our web browsers make are HTTPS. Every one of those connections returns a certificate. And if that site's web browser has not done the courtesy of stapling a valid OCSP certificate to its own certificate, and if the connecting client has been configured to use OCSP, then without allowing for some reasonable local caching, every one of those received HTTPS certificates not having a stapled OCSP response would need to be checked against their respective Certificate Authority's OCSP servers every single time. So it's not just web servers that are retrieving and caching OCSP certificates, it's anyone who's relying upon a web server certificate that does not include a stapled response.

Put another way, if OCSP responses could not be cached for some reasonable compromise time, the entire world would be continuously DDoSing the world's Certificate Authorities' OCSP services due to the need to always look up the latest status of every unstapled certificate they had received. And if you think about it, stapling doesn't solve the problem either. It just moves it to the server-side because, if web servers were not similarly able to staple their most recently received cached and still valid responses, that is, if web servers could not cache the response, then every web server would be pounding away at their Certificate Authority's OCSP service for every new certificate they were sending for every new TLS connection they accepted. So it's clear that caching of OCSP status is crucial for the survival of the system. Which brings us to how long that caching should be.

I mentioned that the Certificate Authorities of every site I checked were issuing OCSP responses with a seven-day lifetime. There's a terrific clean and simple online service for displaying OCSP responses. It's at https://certificatetools.com/ocsp-checker, and I gave it an easier-to-use GRC shortcut of OCSP. So you can just go to grc.sc/ocsp, which will bounce your browser over to the OCSP checker at the certificatetools.com site. If you put whatever domain you like in there, you'll receive the OCSP response from that site's Certificate Authority. Just as I was assembling this, I thought to put in "TWiT.tv," and I was rewarded with the shortest OCSP lifetime I had encountered so far. TWiT.tv's Certificate Authority is GoDaddy, and GoDaddy's OCSP expiration is only four days. I haven't seen anything that short anywhere else. But the standard everywhere else is seven days, Google and everyone.

What we just saw with my "revoked.grc.com" site is a perfect model for malicious exploitation. If a bad guy manages to get hold of an important website certificate and sets up a malicious clone of the site using that certificate, they will definitely want their web server to be stapling valid OCSP responses to every connection since doing so prevents the client from looking any further. But the moment that certificate's OCSP response changes to show that its certificate has been revoked, the malicious server will want to continue using the last good OCSP response it had received for as long as it lasts, which might give it another few days of malicious use before its site spoofing would be brought to a close.

Thus the compromise. It's not a solution that immediately detects certificate revocation, but it's sure better than it was 10 years ago when we looked at this back in 2014, and GRC's revoked site with its revoked certificate was just sitting there for years without any regularly configured browser even noticing. That is not so today. Now we know that within typically six days, seven at the most, if a certificate is revoked, the OCSP for that certificate will either not be signed, not be stapled by a malicious server, in which case maybe the user's browser will go out and ask for an OCSP verification on its own - we don't know what it would do. And I'm kind of curious. But I'm not pursuing it because, as they say, "Wait, there's more."

Believe it or not, three weeks ago today, on July 23rd, Let's Encrypt, which currently commands nearly 60% of the global web browser TLS certificate space, posted their news under the headline "Intent to End OCSP Service." I kid you not. Here's what Let's Encrypt posted. They said: "Today" - three weeks ago; right? Three weeks ago today they said: "Today we are announcing our intent to end Online Certificate Status Protocol support in favor of Certificate Revocation Lists (CRLs) as soon as possible."

Okay. What? We already had Certificate Revocation Lists. They were much worse, a total disaster, in fact. Which is why OCSP was created. We're going back to that? Why? They wrote: "OCSP and CRLs are both mechanisms by which CAs can communicate certificate revocation information, but CRLs have significant advantages over OCSP." Okay. "Let's Encrypt has been providing an OCSP responder since our launch nearly 10 years ago. We added support for CRLs two years ago, in 2022. Websites and people who visit them will not be affected by this change, but some non-browser software might be. We plan to end support for OCSP primarily because it represents a considerable risk to privacy on the Internet."

Okay. So they're talking about non-stapled OCSP, where everybody is making queries, their own queries, rather than relying upon the certificate being stapled to and accompanying the cert from the web server. Privacy. And they're right, that's a problem. They said: "When someone visits a website using a browser or other software that checks for certificate revocation via OCSP, the Certificate Authority operating the OCSP responder immediately becomes aware of which website is being visited from that visitor's particular IP address. Even when a CA intentionally does not retain this information, as is the case with Let's Encrypt, CAs could be legally compelled to collect it. CRLs do not have this issue."

Okay. So again, I ask myself, has Let's Encrypt not heard of OCSP stapling, which solves this problem? Since they must have, it must be the case that, since OCSP stapling is not mandatory, it's just polite, even today not all web servers are going to the trouble of stapling. Okay. Anyway, let's hear from Let's Encrypt.

They continue: "We are also taking this step because keeping our CA infrastructure as simple as possible is critical for the continuity of compliance, reliability, and efficiency at Let's Encrypt. For every year that we've existed, operating OCSP services has taken up considerable resources that can soon be better spent on other aspects of our operations. Now that we support CRLs, our OCSP service has become unnecessary." Wow.

"In August of 2023" - so exactly one year ago - "the CA/Browser Forum passed a ballot to make providing OCSP services optional for publicly trusted CAs like Let's Encrypt. With one exception, Microsoft, the root programs themselves no longer require OCSP. As soon as the Microsoft Root Program also makes OCSP optional, which we are optimistic will happen within the next six to 12 months" - okay, and that's from this posting, which was three weeks ago - "Let's Encrypt intends to announce a specific and rapid timeline for shutting down our OCSP services." This is 60% of the certificate space. "We hope to serve our last OCSP response between three to six months after that announcement. The best way to stay apprised of updates on these plans is to subscribe to our API Announcements category on Discourse.

"We recommend that anyone relying on OCSP services today start the process of ending that reliance as soon as possible. If you use Let's Encrypt certificates to secure non-browser communications such as a VPN, you should ensure that your software operates correctly if certificates contain no OCSP URL. Fortunately, most OCSP implementations 'fail open,' which means that an inability to fetch an OCSP response will not break the system." And so that answers the question I had about what browsers would do. Do they look it up on their own if the OCSP response is not stapled, or do they just shrug and let the user use it anyway, which apparently is what they do.

Okay. So in order to figure out what was going on, I referred to their statement, "We added support for CRLs in 2022," and I found their news about that. Here's what Let's Encrypt posted nearly two years ago on September 7th, 2022 under their headline "A New Life for Certificate Revocation Lists (CRLs)." They said: "This month, Let's Encrypt is turning on new infrastructure to support revoking certificates via Certificate Revocation Lists. Despite having been largely supplanted by the Online Certificate Status Protocol over a decade ago, CRLs are gaining new life

with recent browser updates." What? "By collecting and summarizing CRLs for their users, browsers are making reliable revocation of certificates a reality, improving both security and privacy on the web.

"Let's talk about exactly what this new infrastructure does, and why it's important. When a certificate becomes untrustworthy, for instance because its private key was compromised, that certificate must be revoked and that information publicized so that no one relies upon it in the future. However, it's a well-worn adage in the world of the Web Public Key Infrastructure, the Web PKI, that revocation is broken. Over the history of the Web PKI, there have been two primary mechanisms for declaring that a TLS/SSL certificate should no longer be trusted: Certificate Revocation Lists and the Online Certificate Status Protocol (OCSP). Unfortunately, both have major drawbacks.

"CRLs are basically just lists of all of the certificates that a given Certificate Authority has issued which have been revoked. This means that they're often very large, easily the size of a whole movie. It's inefficient for your browser to download a giant list of revoked certificates just to check if the single certificate for the site you're visiting right now is revoked. These slow downloads and checks made web page loads slow, so OCSP was developed as an alternative."

They say: "OCSP is sort of like 'what if there were a separate CRL for every single certificate?' When you want to check whether a given certificate has been revoked, your browser can check the status for just that one certificate by contacting the CA's OCSP service. But because OCSP infrastructure has to be running constantly and can suffer downtime just like any other web service, most browsers treat getting no response at all as equivalent to getting a 'not revoked' response. This means that attackers can prevent you from discovering that a certificate has been revoked simply by blocking all your requests for OCSP information.

"To help reduce load on the CA's OCSP services, OCSP responses are valid and can be cached for about a week. But this means that clients don't retrieve updates very frequently, and often continue to trust certificates for a week after they're revoked. And perhaps worst of all, because your browser makes an OCSP request for every website you visit, a malicious or legally compelled CA could track your browser behavior by keeping track of what sites you request OCSP for.

"So both of the existing solutions don't really work. CRLs are so inefficient that most browsers don't check them, and OCSP is so unreliable that most browsers don't check it. We need something better. One possible solution that's been making headway recently is the idea of proprietary, browser-specific CRLs. Although different browsers are implementing this differently - for example, Mozilla calls theirs CRLite, and Chrome's are CRLSets - the basic idea is the same. Rather than having each user's browser download large CRLs when they want to check revocation, the browser vendor downloads the CRLs centrally. They process the CRLs into a smaller format such as a Bloom filter, then push the new compressed object to all of the installed browser instances using pre-existing rapid update mechanisms. Firefox, for example, is pushing updates as quickly as every six hours.

"This means that browsers can download revocation lists ahead of time, keeping page loads fast and mitigating the worst problems of vanilla CRLs. It keeps revocation checks local, and the pushed updates can take immediate effect without waiting for a potentially week-long OCSP cache to expire, preventing all the worst problems with OCSP.

"Thanks to the promise of these browser-summarized CRLs, both the Apple and Mozilla root programs are requiring that all CAs begin issuing CRLs before October 1st, 2022." Okay, so that's a year and a half ago, almost two years ago. "Specifically, they are requiring that CAs begin issuing one or more CRLs which together cover all certificates issued by the CA, and that the list of URLs pointing to those CRLs be disclosed in the Common CA Database. This will allow Safari and Firefox to switch to using browser-summarized CRL checking for revocation."

And finally: "When Let's Encrypt was founded, we made an explicit decision to only support OCSP and not produce CRLs at all. This was because the root program requirements at the time only mandated OCSP, and maintaining both revocation mechanisms would have increased the number of places where a bug could lead to a compliance incident.

"When we set out to develop CRL infrastructure, we knew we needed to build for scale, and do so in a way that reflects our emphasis on efficiency and simplicity. Over the last few months we have developed a few new pieces of infrastructure to enable us to publish CRLs in compliance with the upcoming requirements. Each component is lightweight, dedicated to doing a single task and doing it well, and will be able to scale well past our current needs.

"Let's Encrypt currently has over 200 million active certificates on any given day. If we had an incident where we needed to revoke every single one of those certificates at the same time, the resulting CRL would be over 8GB. In order to make things less unwieldy, we will be dividing our CRLs into 128 shards, each topping out at a worst-case maximum of 70MB. We use some carefully constructed math to ensure that, as long as the number of shards does not change, all certificates will remain within the same shards when the CRLs are re-issued, so that each shard can be treated as a mini-CRL with a consistent scope."

Okay. So they weren't explicit about this, but I assume their reason for breaking their master CRL into multiple consistent mini-CRLs is that any addition of a certificate to a CRL would then affect only one of the 128 individual CRLs. So that rather than replacing the entire list, only 1/128th of the entire list would require updating.

And they said: "As part of developing these new capabilities, we have also made several improvements to the Go standard library's implementation of CRL generation and parsing. We look forward to contributing more improvements as we and the rest of the Go community work with CRLs more frequently in the future. Although we will be producing CRLs which cover all certificates that we issue, we will not be including those URLs in the CRL Distribution Point extension of our certificates." In other words, they're going to be creating CRLs, Certificate Revocation Lists, but they're not going to be making their access public.

They said: "For now, as required by the Baseline Requirements, our certificates will continue to include an OCSP URL which can be used by anyone to obtain revocation information for each certificate. Our new CRL URLs will be disclosed only in the CCADB" - that's the CA/Browser database - "so that the Apple and Mozilla root programs can consume them without exposing them to potentially large download traffic from the rest of the Internet at large." In other words, this new model would be that Apple will obtain the CRL from Let's Encrypt. Mozilla will obtain the CRL from Let's Encrypt. They will then process them and periodically, when necessary, upload them to our browser instances so our browsers are always maintaining a relatively recent CRL, you know, master CRL.

And they finish: "There's still a long way to go before revocation in the Web PKI is truly fixed." In other words, it never has been. They wrote: "The privacy concerns around OCSP will only be mitigated once all clients have stopped relying upon it, and we still need to develop good ways for non-browser clients to reliably check revocation information. We look forward to continuing to work with the rest of the Web PKI community to make revocation checking private, reliable, and efficient for everyone."

Now, digging around in the CA/Browser Forum history, I found ballot measure SC-063 from a year ago. It was titled - and I'm not going to share the whole thing. I'll just summarize. It was titled "Make OCSP Optional, Require CRLs, and Incentivize Automation." In other words, we're switching back. We're giving up everything, we're shutting down OCSP, we're going back to CRLs, but this time we're going to do it differently.

Okay. The result of voting on this measure - I checked the ballot results. The result of voting on this measure to do that was 23 certificate issuers voting yes, and only one certificate issuer voting no. And all three of the certificate consumers who were participating in the Forum, namely Google, Mozilla, and Apple, voted yes, with no certificate consumer voting no. So it's very clear that the certificate revocation tides will be turning once again in an attempt to continue fixing what everyone agrees is a barely functioning certificate revocation system.

As I was reading through the Let's Encrypt plans, a couple of things occurred to me. As we all know, certificates self-expire. This is highly significant for the practicality of certificate revocation lists since these lists only need to suppress the trust of certificates that have not yet expired. In other words, all certificates will eventually become untrusted by virtue of their own "not valid

after" timestamps. Ten years ago, web browser certificates were available with five-year lifetimes. This meant that a certificate that escaped the control of its owner, or was subjected to a Heartbleed attack or whatever, or may have been misused, would need to remain on its CA's CRL for the balance of its life, which might be as long as five years. Therefore, the gradual reduction in certificate lifetime that the industry has been facilitating over the past 10 years has significantly reduced the CRL burden.

With all TLS web certs now expiring after a maximum of 398 days, CRLs can be much shorter and smaller because the certificate will, you know, a bad certificate will take itself out after a maximum of 398 days. Then it can be dropped from the CRL. And despite Let's Encrypt having nearly 60% of the certificate market, their much shorter 90-day certificates which are being issued through their ACME automation means that their CRLs can benefit from their even shorter lifetimes.

The other thing that has changed in the past 10 years is the general increase in bandwidth and local storage capacity, all while reducing cost. Right? This makes shipping web browser updates far more practical and effectively invisible to their users. Today's web browsers have become large operating systems in their own right, yet most of today's users aren't even aware as these behemoths are being updated silently in the background.

This all suggests that the move away from individual certificate revocation checking to browser-side list checking can make sense. It does impose a new burden upon all browser vendors since they now will need to be continuously collecting and merging the individual CRLs from every source of web browser certificates, and as we've seen in the past there are a great many of those.

So we may not have yet seen the end of GRC's revoked.grc.com site. In another year or two, once the switch has been made from OCSP back to browser-based Certificate Revocation Lists, I think it's going to be fun and interesting to create a new freshly revoked certificate for that site and see what happens, see how it goes.

Leo: This is a tough computer science problem.

Steve: Yes.

Leo: Doesn't look like we have the answer to it.

Steve: Yes. That is exactly right. It is, you know, the idea is that certificates are meant to be freestanding representations of something.

Leo: Right.

Steve: That have a lifetime. Yet that representation may, you know, may not - or the representation may outlive the truth of what it is saying. So you need some way of saying, uh, whoops, we did promise that, and we signed it. And we were right when we said it, but something happened. Whoops. Yeah, Heartbleed or bad guys or whatever.

Leo: Well, even if you divide it up into 128 70MB files, that's still a lot of data.

Steve: Well, that was worst-case.

Leo: Okay.

Steve: They were saying, if they had to replace - I think that was if they had to replace every single certificate all at once. And that's Let's Encrypt with 60% of the certificate market. Yeah, so they were taking an 8GB file, dividing it into 128 pieces. So I think what's going to happen is we're all going to have local CRLs of all of the various CAs' revoked and not-yet-expired certificates. And boy, Leo, are we going to have fun when that happens.

Leo: That seems like a lot of data still.

Steve: Yeah. What I'm going to do, I was thinking about this actually, somehow I was able to think while I was reading at the same time, which sort of surprised me. I didn't think I was able to do that.

Leo: I can't even think while I'm switching cameras, so you're way ahead of me.

Steve: I was thinking I'm going to put DigiCert's OCSP site into my hosts file so that I can blackhole it. That way my server will not be able to staple. And then I'm going to get a certificate again and revoke it, which it won't be able to staple, and we're going to find out what our browsers do once stapling is suppressed.

Leo: Well, they're probably going to fail insecure. Isn't that what they've been doing?

Steve: I think they're going to fail open, yup.

Leo: Yeah, fail open.

Jump To Top