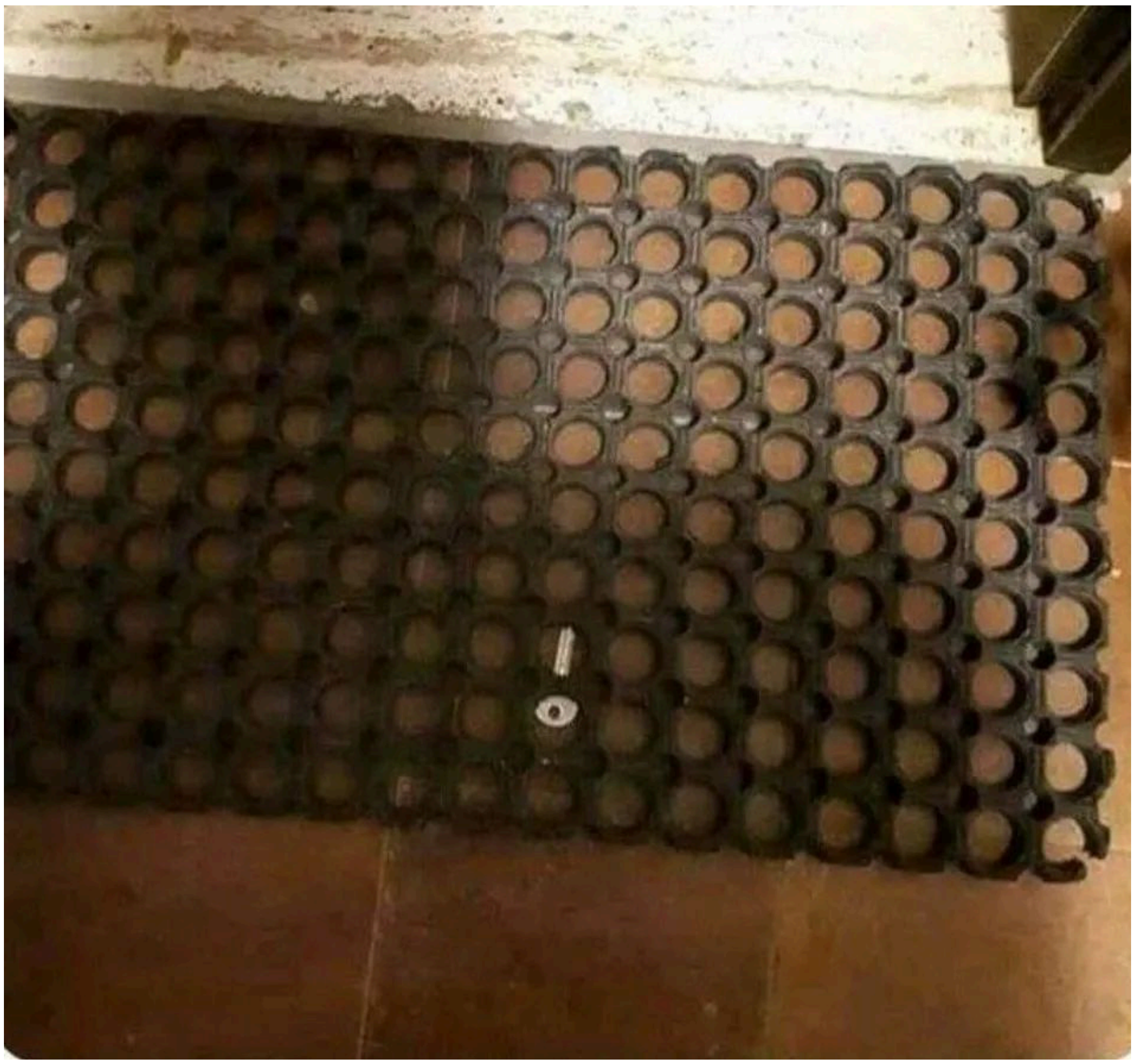
Security Now! #987 - 08-13-24
Rethinking Revocation

This week on Security Now!

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. And believe it or not, the entire certificate revocation system that the industry has just spent the past ten years getting working is about to be scrapped in favor of what never worked before. Go figure.

Isn't the expression *"Hiding the key under the mat?"*



Security News

Sitting Ducks DNS attack

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. This generally doesn't happen with major sites, since the problem would be seen and resolved almost immediately. But many large organizations register every typo variation of their primary domain name in order to keep so-called "typo squatters" from registering those domains and catching people who mistype the actual domain name. Since those domains are of far lesser value, their hijacking can go unnoticed.

The most recent round of this DNS hijacking surfaced recently when the two security firms Eclypsium and InfoBlox co-published their discovery. As I said, though, it should really be considered a rediscovery, which is not to take anything away from them, 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 it right. Eclypsium's write-up said:

> *Multiple threat actors have been exploiting this attack vector which we are calling Sitting Ducks since at least 2019 to perform malware delivery, phishing, brand impersonation, and data exfiltration. As of the time of writing, numerous DNS providers enable this through weak or nonexistent verification of domain ownership for a given account. There are an estimated one million exploitable domains and we have confirmed more than 30,000 hijacked domains since 2019. Researchers at Infoblox and Eclypsium, who discovered this issue, have been coordinating with law enforcement and national CERTs since discovery in June 2024.*

Again, I'm glad this has resurfaced. But we talked about this same issue eight years ago when on December 5th, 2016, security researcher Matthew Bryant posted his discovery under the title: "The Orphaned Internet – Taking Over 120K Domains via a DNS Vulnerability in AWS, Google Cloud, Rackspace and Digital Ocean." This was Matthew's second of two postings about this problem where he writes:

> *Recently, I found that Digital Ocean suffered from a security vulnerability in their domain import system which allowed for the takeover of 20K domain names. If you haven't given that post a read 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 verification of ownership of the domain name itself. 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 to control it. In many cloud providers this means that anyone can create a DNS zone for that domain and take full control over the domain. This allows an*

attacker to take full control over the domain to set up a website, issue SSL/TLS certificates, host email, etc. Worse yet, after combining the results from the various providers affected by this problem over 120,000 domains were vulnerable (likely many more).

Reportedly, Russian hackers have been doing this for years. This time around Eclypsium and Infoblox have brought this to everyone's attention again, and giving it a catchy name the "Sitting Duck" attack. I particularly like this name since it nicely and accurately captures the essence, which is that a surprising number of domains are, in fact, sitting ducks. The naming was a bit of a stretch, but it's fun. The initials DNS are repurposed to "Ducks Now Sitting" – thus, sitting ducks.

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 in Matthew Bryant's original posting he explained the entire problem and provided means for its detection. He also responsibly disclosed this discovery to a bunch of cloud providers where, due to the nature of delegating DNS to them the trouble is or was rampant.

Google Cloud DNS had around 2500 domains affected. They did what they could. Amazon Web Services Route 53 DNS had around 54,000 domains affected so they performed multiple mitigations for the problem. Digital Ocean had around 20,000 domains at risk and they worked to fix what they could. And, interestingly, Rackspace felt differently than others about the problem. They had around 44,000 domains affected and said that this was not their problem.

Matthew explains how simple it is to take over an abandoned Rackspace domain:

Rackspace offers a Cloud DNS service which is included free with every Rackspace account. Unlike Google Cloud and AWS Route53 there are only two nameservers (dns1.stabletransit.com and dns2.stabletransit.com) for their cloud DNS offering so no complicated zone creation/deletion process is needed. 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:

- *Under the Cloud DNS panel, click the "Create Domain" button and specify the vulnerable domain and a contact email and TTL.*
- *Now simple create whatever DNS records you'd like for the taken over domain.*

This can be done for any of the 44K domain names to take them over. Rackspace does not appear to be interested in patching this (see below) so if you are a Rackspace customer please ensure you're properly removing Rackspace's nameservers from your domain after you move your domain.

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. 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.

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."*

If this sounds vaguely familiar it may be because we recently talked about that somewhat similar DNS record problem where an Azure subdomain of Microsoft's was decommissioned but its DNS record was never removed, thus allowing bad guys to register the subdomain and obtain its traffic. There are many ways for DNS to go wrong. Essentially, it's all about pointers... and we know what a problem people have with pointers!

During this round of rediscovery of this ago-old problem which has been exacerbated by the move to the cloud, Brian Krebs interviewed Steve Job, the founder and senior vice president of the firm DNSMadeEasy by email. Steve Job said that the problem isn't really his company's to solve, 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 here's the takeaway from all of this. Despite all the press it has received, largely because they gave it a great name this time around, as we've seen, the "Sitting Ducks" problem is not new. The registration of a domain name contains pointers to the domain's DNS nameservers for the 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 the cloud and start receiving all of its traffic.

As I noted before, this is unlikely to be happening with anyone's primary accounts since those accounts would be well watched. But DNS for low-priority domains such as typo-squatting prevention accounts which might have a server in the cloud used to bounce incoming traffic over to the correct domain would be … well … sitting ducks.

A Bad RCE in another Microsoft server

When you're a Microsoft shop, headlines which read *"Exploitable PoC released for a 0-Click RCE that threatens all Windows servers"* your heart skips a beat. And if your organization is publicly exposing the Windows Remote Desktop Licensing Service (RDL) then it's hopefully not too late for you to do something about it.

Three security researchers have detailed and published proof-of-concept (PoC) exploit code for a critical CVSS 9.8 vulnerability being tracked as CVE-2024-38077 which is referred to as "MadLicense." This 9.8'er impacts **all** iterations of Windows Server from 2000 to 2025 – in other words, all of them. And if your service is publicly exposed it'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 0-click and affects all Windows services which are exposing the Windows Remote DesktopLicensing Service.

The Windows Remote Desktop Licensing Service is responsible for managing licenses for Remote Desktop Services and it is consequently deployed across many organizations. Security researchers have identified a minimum of 170,000 RDL services are directly exposed to the internet which renders them all readily susceptible to exploitation. And the nature of the exposed service means that these are not going to be small enterprises. These are going to be tasty targets.

This so-called "MadLicense" vulnerability arises from a simple heap overflow. By manipulating user-controllable 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 (POC) exploit on Windows Server 2025, achieving a near 100% success rate. The exploit effectively circumvents all contemporary mitigations.

While the PoC demonstrated the vulnerability's exploitation on Windows Server 2025, the researchers emphasized that the bug could be exploited more quickly and efficiently on older versions of Windows Server, where fewer mitigations are in place. The PoC was designed to load a remote DLL – Yikes – but the researchers noted that with slight modifications, it could execute arbitrary shellcode within the RDL process, making the attack even stealthier – meaning that the infected system would not be reaching out onto the Internet to obtain that malicious DLL.

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, the fact that it was initially marked as "exploitation less likely" highlights the potential for underestimating such threats.

So anyone who is not actively using and depending upon this Remote Desktop Licensing Service should definitely shut it down immediately. And if for any reason you're a month behind on updates (since today is August's Patch Tuesday), you should seriously consider at least applying the patch for this vulnerability. Now that a proof of concept has been posted on Github we know it won't be long before the bad guys are attacking any unpatched network.

SinkClose

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 after DigiCert's mass revocation event and my own "revoked.grc.com" server. It turns out that in researching this further I've found that everything we've been doing for certificate revocation over the past ten years is about to change again. So we'll get there at the end of today's discussion.

For now, let's talk about the big runner up for this week, which was dubbed "SinkClose" by the two security researchers who discovered it. We've just been talking about SecureBoot and how AMI's never-meant-to-be-shipped sample platform key was indeed shipped in as many as 850 different makes and models of PCs. That's not good. But another thing that's not good is when two researchers discover a fundamental vulnerability in AMD's processors – and I mean nearly all of them – which allows for the subversion of this same SecureBoot process regardless of what key its platform is using.

WIRED's coverage of this discovery carried the headline: "'Sinkclose' Flaw in Hundreds of Millions of AMD Chips Allows Deep, Virtually Unfixable Infections." Now, in fairness, we seem to be encountering similar "sky is falling" headlines almost weekly. So what's going on here? WIRED's subhead is not any more encouraging. It reads: "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. I'm going to first share a somewhat trimmed version of what WIRED wrote then we'll see where we are:

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.

At the DEFCON hacker conference, researchers from the security firm IOActive, 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, 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 even earlier.

The researchers, 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, 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 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 the 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.*

Okupski said: "Imagine nation-state hackers or whoever wants to persist on 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 SPI Flash programmer and meticulously scouring the memory would allow the malware to be removed."

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, 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." (The term "embedded," in this case, refers to AMD chips found in systems such as industrial devices and cars.) For its EPYC processors designed for use in data-center servers, specifically, the company noted that it released patches earlier this year. 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.

In a background statement to WIRED, AMD emphasized the difficulty of exploiting Sinkclose: 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 vault door.

Nissim and Okupski respond that while exploiting Sinkclose requires kernel-level access to a machine, such vulnerabilities are exposed in Windows and Linux practically every month. 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."

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 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. 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 SMM level.

Okupski said: "I think it's the most complex bug I've ever exploited."

Nissim and Okupski, both of whom specialize in the security of low-level code like processor firmware, say they first decided to investigate AMD's architecture two years ago, simply because they felt it hadn't gotten enough scrutiny compared to Intel, even as its market share rose. They found the critical TClose edge case that enabled Sinkclose, they say, just by reading and rereading AMD's documentation. "I think I read the page where the vulnerability was about a thousand times," says Nissim. "And then on one thousand and one, I noticed it." They alerted AMD to the flaw in October of last year but have waited nearly 10 months to give AMD more time to prepare a fix.

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 months to come, 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, it shouldn't prevent users from patching as soon as possible. Sophisticated hackers may already have discovered their technique—or may figure out how to after Nissim and Okupski present their findings at DEFCON.

Even if Sinkclose requires relatively deep access, the IOActive researchers warn, the far deeper level of control it offers means that potential targets shouldn't wait to implement any fix available. "If the foundation is broken," says Nissim, "then the security for the whole system is broken."

https://www.amd.com/en/resources/product-security/bulletin/amd-sb-7014.html

I have a link in the show notes for anyone wishing for additional information from AMD. AMD refers to this as an SMM (System Management Mode) Lock Bypass and AMD's one sentence summary is *"Researchers from IOActive have reported that it may be possible for an attacker with ring 0 access to modify the configuration of System Management Mode (SMM) even when SMM Lock is enabled."*

As we know, 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. But we may need to wait a few months until the IOActive guys are able to disclose more and having a definitive test for the presence of the patch or benign test for the vulnerability would be very useful.

The CLFS.SYS BSoD

We have yet another example of Microsoft apparently choosing not to fix what appears to be a somewhat significant problem. Allowing anyone to crash any recent and fully patched Windows system would seem to 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 (CLFS) driver can instantly trigger the infamous blue screen of death across any recent versions of Windows. CLFS is a user- and kernel-mode logging service that helps applications record and manage logs. It's also a popular target for hacking.

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 (PoC) 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 I crashed my server."

The underlying issue — labeled CVE-2024-6768 — concerns base log files (BLFs), a type of CLFS file that contains metadata used for managing logs. The CLFS.sys driver, it seems, does not adequately validate the size of data within a particular field in the BLF. Any attacker with access to a Windows system can craft a file with incorrect size information which will confuse the driver. Then, unable to resolve the inconsistency, it triggers KeBugCheckEx, the function that triggers a blue screen crash.

The CVE has earned a "medium" 6.8 out of 10 score on the CVSS scale. It doesn't affect the integrity or confidentiality of data, nor cause any kind of unauthorized system control. 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 can be paired with other exploits to 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 back up and it's fine now, but that might have been somebody hiding their activity — hiding the fact that they wanted it to reboot so that a new setting would take effect."

Fortra first reported its findings last Dec. 20. After months of back and forth, they said that Microsoft closed their investigation without acknowledging it as a vulnerability or applying a fix. Thus, as of this writing, it persists in Windows systems no matter how updated they are.

In recent weeks, Windows Defender has been identifying Fortra's proof of concept as malware. But besides running Windows Defender and trying to avoid running any binary that exploits it, there's nothing organizations can do to deal with CVE-2024-6768 until Microsoft releases a patch. Dark Reading has reached out to Microsoft for its input on CVE-2024-6768.

So get a load of this timeline which Fortra posted on their vulnerability disclosure report:

- December 20, 2023 – Reported to Microsoft with a Proof-of-Concept exploit.
- January 8, 2024 – Microsoft responded that their engineers could not reproduce the vulnerability.
- January 12, 2024 – Fortra provided a screenshot showing a version of Windows running the January Patch Tuesday updates and a memory dump of the crash.
- February 21, 2024 – Microsoft replied that they still could not reproduce the issue and they were closing the case.
- February 28, 2024 – Fortra reproduced the issue again with the February Patch Tuesday updates installed and provided additional evidence, including a video of the crash condition.
- June 19, 2024 – Fortra followed up to say that we intended to pursue a CVE and publish our research.
- July 16, 2024 – Fortra shared that it had reserved CVE-2024-6768 and would be publishing soon.
- August 8, 2024 – Reproduced on latest updates (July 2024 Patch Tuesday) of Windows 11 and Server 2022 to produce screenshots to share with media.
- August 12, 2024 – CVE publication date.

And this is the same company that wants to continuously record everything every Windows PC does while assuring us that our entire recorded history will be perfectly safe.   No thanks.

IsBootSecure

Two weeks ago we covered the news from Binarly that an estimated 10% of PCs had shipped with, and were currently using and replying upon AMI's sample demonstration UEFI Platform Keys which had never been intended to be shipped into production. And at least some of those were known to have been publicly leaked online which rendered these keys vulnerable to attack by the very boot-time malware that the entire "SecureBoot" system was designed to prevent. Binarly named their discovery "PKfail". The following Friday, after seeing many people wanting to know about their own machines and that there was not any straightforward and foolproof way to see a machine's keys, I decided that a new piece of GRC freeware would be needed.

Last Tuesday I introduced the "IsBootSecure" freeware which was, at the time, a bare bones Windows console app. On Sunday I updated that to a regular more user-friendly graphical dialog app and I fleshed out its supporting webpage. In some testing yesterday a couple of minor cosmetic problems were found but I decided not to withhold the app's first release. After today's podcast I'll fix those and send out an email to everyone who has subscribed to GRC's software and services news email.

Rethinking Revocation

Last week we covered DigiCert's immediate revocation of more than 83 thousand certificates after it was discovered that their automation software had been found not to 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.

At the end of that discussion we talked about the historical mess of certificate revocation and I reminded everyone about GRC's "revoked.grc.com" test website which I originally created back in 2014 when the podcast covered this topic in depth. 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" webserver. And as we all know, five days later on August 6th during last week's podcast, Leo and I and all of our online listeners 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 verified was revoked, was still being happily displayed by everyone's browsers... 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 me that none of their various web browsers would display the revoked.grc.com site. It was showing as an untrusted site with a revoked certificate. So what happened.

What happened is that five days before, 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. It was and the page came right up. 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 at: http://ocsp.digicert.com
(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.)

Upon finding that URL, 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, 7-day, timestamped certificate attesting to the fact that this bright and shiny new certificate was as 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 an 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.

Having seen that everything looked good and was working as 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 the 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 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 certificates 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.

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 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 here.

Then 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 OCSP update. 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 and are certain to obtain a replacement from its certificate authority well before the current certificate expires. And at that point, six days after the certificate’s creation and its almost immediate revocation, 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.

To see what I mean by that, let’s turn the clock back ten years to see how we got here. We first looked at all of this ten years ago back in 2014. 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 ten years ago but their services were not used by default. As we know, 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. This allows anyone who receives a certificate that the CA signed to obtain the URL from within the certificate and query the CA’s OCSP service right then and there to obtain up-to-to-moment information about any specific domain’s certificate. What the OCSP service returns is a timestamped, short-lived and signed-by-the-CA attestation of the current status of the queried certificate. It created, as its name says, an Online Certificate Status Protocol.

At the time we played with Firefox's settings, which could cause Firefox to query a CA's OCSP servers in real time. But there were a couple of problems with this. One problem was that OCSP services were still not very reliable and could be slow to respond. So a browser visits website, receives the site's 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. So this can 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? Does the web browser refuse to connect to a site that it cannot positively verify?

The goal is to never trust an untrustworthy website. 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 CA's OCSP service doesn't respond or might be very slow. As a consequence, ten 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 pretty much frantic about the performance of their web browser. The last thing they were going to do was anything that might slow it down in any way.

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 – could track the sites that users were visiting since 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.

The reason I was so excited about OCSP stapling ten 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 its certificate's CA to the certificate it's sending to the browser. Now the web browser doesn't need to perform another query and fetch. So there's no delay, no extra fetching and waiting, and no privacy issue. Essentially the server is saying: "Here's a certificate my certificate authority issued up to a year ago... **and** here's a much more recent signed assertion from the same authority stating that the certificate is still in good standing."

What we learned over the past week is that the phrase "much more recent" might not always be as recent as we would like. The OCSP response for every domain I have checked, mine, Google, Let's Encrypt, Hover, Facebook (actually Facebook also uses DigiCert so GRC is using the same OCSP service as Facebook) – they are all issuing OCSP response certificates with a 7-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 our web browsers make is 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 is relying upon a web server certificate that doesn't include a stapled OCSP 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 Authority's 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 OCSP responses, 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 7-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 goto https://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 quickly 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've encountered so far. TWiT.tv's certificate authority is GoDaddy and GoDaddy's OCSP expiration is only 4 days. I haven't seen anything that short anywhere else.

But the standard everywhere else is 7 days. 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 – which might give it another week 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 ten 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.

But 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 brows er TLS certificate market, posted their news: “Intent to End OCSP Service” (I kid you not.)

Here’s what Let’s Encrypt posted:

> *Today we are announcing our intent to end Online Certificate Status Protocol (OCSP) support in favor of Certificate Revocation Lists (CRLs) as soon as possible.*

What?!?!?! We already had Certificate Revocation Lists. They were much worse — a total disaster! We’re going back to that? Why??

> *OCSP and CRLs are both mechanisms by which CAs can communicate certificate revocation information, but CRLs have significant advantages over OCSP. Let’s Encrypt has been providing an OCSP responder since our launch nearly ten 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. When someone visits a website using a browser or other software that checks for certificate revocation via OCSP, the Certificate Authority (CA) 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.*

Wait... 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. Anyway, let’s hear from Let’s Encrypt:

> *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 have 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.*
>
> *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 twelve months* [from this posting three weeks ago]*, Let’s Encrypt intends to announce a specific and rapid timeline for shutting down our OCSP services. We hope to serve our last OCSP response between three and 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.

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 the headline "A New Life for Certificate Revocation Lists":

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 for over a decade now, CRLs are gaining new life with recent browser updates. 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 (CRLs) 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 (CA) 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.

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 of your requests for OCSP information. To help reduce load on a 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 browsing 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 has been making headway recently is the idea of proprietary, browser-specific CRLs. Although different browsers are implementing this differently (e.g. 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 6 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 of 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. Specifically, they are requiring that CAs begin issuing one or more CRLs which together cover all certificates issued by that CA, and that the list of URLs pointing to those CRLs be disclosed in the Common CA Database (CCADB). This will allow Safari and Firefox to switch to using browser- summarized CRL checking for revocation.

Our New Infrastructure

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 8 gigabytes. 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 70 megabytes. We use some carefully constructed math to ensure that – as long as the number of shards doesn't change – all certificates will remain within their same shards when the CRLs are re-issued, so that each shard can be treated as a mini-CRL with a consistent scope.

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 only affect one of the 128 individual CRLs. So rather than replacing the entire list, only 1/128th of the entire list would require updating.

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. 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 CCADB, 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.

There's still a long way to go before revocation in the Web PKI is truly fixed. The privacy concerns around OCSP will only be mitigated once all clients have stopped relying on 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.

Digging around in the CA/Browser forum history, I found ballot measure SC-063, from a year ago, titled "Make OCSP Optional, Require CRLs, and Incentivize Automation". The result of voting on this measure was 23 certificate issuers voting YES and only one voting no. And all three certificate consumers participating in the Forum: Google, Mozilla and Apple voting 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 5-year lifetimes. This meant that a certificate that escaped the control of its owner, or was subjected to Heartbleed attack, or may have been mis-issued, would need to remain on its certificate authority's CRL for the balance of its life which might be as long as 5 years. Therefore, the gradual reduction in certificate lifetime that the industry has been facilitating has significantly reduced the CRL burden. With all TLS web certs now expiring after 398 days, CRLs can be much shorter and smaller.

And despite Let's Encrypt having nearly 60% of the certificate market, the much shorter 90-day certificates that are 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 ten years is the general increase in bandwidth and local storage capacity, all while reducing cost. 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 checking can make sense. It does impose a new burden upon all browser vendors since they will be needing to continuously collect and merge the individual CRLs from every source of web browser certificates, and as we've seen in the past, there are a great many.

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