Security Now! #740 - 11-12-19

Credential Delegation

This week on Security Now!

This week we check in on the developments of the long-term now-working full consumer jailbreak of iOS devices from the iPhone 4S through the iPhone X. We examine the strange case of the misbehaving transducer, catch up on the rapidly evolving exploitation of the BlueKeep vulnerability, check out Mozilla's rebuttal to COMCAST's attack on DoH, examine the surprising state of web browser support for DoH, remind Linux and BSD users to refresh their distros after an important flaw was disclosed in a widely used archive library. Then we take a deep drive into the operation of a newly announced forthcoming solution and standard for significantly improving TLS website certificate security, known as "TLS Credential Delegation."

Here comes the iOS "EverBreak" !!!

Security News

CheckM8 & Checkra.in moves to first public beta
As our picture of the week shows, early public beta preview of the USB-tethering checkra1n iOS jailbreak based on the unpatchable iOS Checkm8 bootrom exploit started appearing over the weekend. So far we've had v0.9, v0.9.1 and v0.9.2.

The "Official" site is https://checkra.in/ and a note at the bottom of the page states that: "The only official domain to download checkra1n from is https://checkra.in. Please make sure to avoid similar-looking domains as they are often malicious sites."

For those who may have missed episode #736, four weeks ago titled "CheckM8", Checkm8 is an extremely rare unpatchable bootrom exploit that can be used against most generations of iPhones and iPads, from the iPhone 4S with the A5 chip through the iPhone 8 and 10 based on the A11 chip.

So, "Checkra1n" can be used to jailbreak all iOS devices running up to iOS 12.3, when, with the introduction of their A12 chip, Apple discovered and quietly patched their BootROM to close this hole.

checkra1n 0.9.2 beta

This release is an early beta preview and as such should not be installed on a primary device. We strongly recommend proceeding with caution.

What's new
- Fixed an issue where the Apple Watch would not receive notifications while jailbroken
- Improve reliability of entering DFU mode
- Fixed an issue where checkra1n could not be used on macOS 10.10
- This beta adds an option to boot into no-substrate mode. To utilise this functionality, hold the volume-up button when the apple logo appears until the device finishes booting. From there you'll be able to uninstall any tweaks causing you issues, and reboot to get back to a normal jailbroken state.

Unsupported devices
checkra1n will eventually support all devices between the iPhone 5s and the iPhone X, however, this beta lacks support for the following devices:

- iPad Air 2
- iPad 5th Gen
- iPad Pro 1st Gen
- Support for these devices will be added in a later release.

Support for the following devices is experimental, and may require more attempts than usual:
- iPhone 5s
- iPad Mini 2 (con't)

- iPad Mini 3
- iPad Air
- Reliability on these devices will be improved in future releases.

Unsupported platforms
- This beta is only available for macOS. Work is ongoing to support Windows and Linux, which will be added in a later release.

Package managers
- At the moment, checkra1n only supports installing Cydia. Support for other package managers is coming soon, and will not require a checkra1n update.

The case of the misbehaving transducer
In one of the weirder pieces of recent research, from a researcher with Japan's University of Electro-Communication and four researchers at the University of Michigan, we have their paper titled:

"Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems."

Wait... "Laser-Based Audio Injection"?? By definition, a "transducer" is any device that converts energy from one form into another. Typically a transducer converts a signal in one form of energy to a signal in another. So, for example, we all know that a speaker is a transducer that converts electrical current into movement to then reproduce sound. Going in the opposite direction a microphone is a transducer that converts incoming sound into movement and then into electricity. A lightbulb is a transducer that converts current into light, and solar panels are transducers that convert light back into electric current.

Now, we have in the past on this podcast encountered some weird transduction behavior, such as the famous case of shouting at an array of hard drives that increased their read error rate and thus reduced their measured data throughput. And we have new oddball to add...

As they describe it in their 17-page paper: "We have identified a semantic gap between the physics and specifications of MEMS (micro-electro-mechanical systems) microphones, where such microphones unintentionally respond to light as if it was sound. Exploiting this effect, we can inject sound into microphones by simply modulating the amplitude of a laser light."

Think about if someone located hundreds of meters away could tell your Amazon or Google voice assistant to unlock the front door or open the garage door.

In their paper's summary Abstract they write: "We propose a new class of signal injection attacks on microphones based on the photoacoustic effect: converting light to sound using a microphone. We show how an attacker can inject arbitrary audio signals to the target microphone by aiming an amplitude-modulated light at the microphone's aperture. We then proceed to show how this effect leads to a remote voice-command injection attack on voice-controllable systems. Examining various products that use Amazon's Alexa, Apple's Siri, Facebook's Portal, and Google Assistant, we show how to use light to obtain full control over

these devices at distances up to 110 meters and from two separate buildings. Next, we show that user authentication on these devices is often lacking or non-existent, allowing the attacker to use light-injected voice commands to unlock the target's smartlock-protected frontdoors, open garage doors, shop on e-commerce websites at the target's expense, or even locate, unlock and start various vehicles (e.g., Tesla and Ford) that are connected to the target's Google account. Finally, we conclude with possible software and hardware defenses against our attacks.

I'm not going to go into this in any greater detail since everyone gets the idea. Re-expressing this in more classic security terms we would say that the lack of authentication that's present in today's voice command systems represents a rational trade-off between convenience and security when used in the typical environment where it's NOT possible for any random stranger at a distance to inject their own unauthenticated voice commands into those devices. This lack of authentication is obviously a problem when voice commands can be injected at a distance.

So falls into the category of "it's unlikely to be a problem, but future devices whose microphones are inadvertently light sensitive might wish to proactively block incoming illumination, which our current crop of voice command devices clearly lack.

I've included the link to their full research paper for anyone who is interested:
https://lightcommands.com/20191104-Light-Commands.pdf

Following up on both of last week's headline topics...

BlueKeep and Microsoft
Microsoft's security team believes that more destructive BlueKeep attacks are on the horizon and urges users and enterprises to apply patches if they've been lagging. I think we all know that if anyone was going to update any of the more than 700,000 Windows systems which are still exposing vulnerable RDP protocol that's susceptible to BlueKeep takeover, it would have happened months ago. I think that these repeated Microsoft warnings must be for political CYA cover.

As we covered last week so far we're only seeing BlueKeep leveraged to install cryptominers. As ZDNet summed things up in their coverage:

> Many security researchers considered the attacks underwhelming and not living up to the hype that was built around BlueKeep for the past six months.
>
> This was because Microsoft said BlueKeep could be used to build wormable (self-spreading) malware. However, the attacks that happened over the weekend did not deploy malware that could spread on its own.
>
> Instead, attackers scanned the Internet for vulnerable systems and attacked each unpatched system, one at a time, deploying a BlueKeep exploit, and then the cryptocurrency miner.
>
> This was far from the self-spreading malware outbreak that Microsoft said BlueKeep could trigger. Furthermore, in many cases, the BlueKeep exploit failed to work, crashing systems.

But Microsoft says this is just the beginning, and that attackers will eventually refine their attacks, and that the worst is yet to come.

Microsoft said last Friday: "While there have been no other verified attacks involving ransomware or other types of malware as of this writing, the BlueKeep exploit will likely be used to deliver payloads more impactful and damaging than coin miners. We cannot discount enhancements that will likely result in more effective attacks."

So that's Microsoft's third warning this year... and I suppose against this backdrop it's not difficult to understand why Microsoft has put a clear limit on the length of time Windows 10 updates can be deferred.

Microsoft also said: "Customers are encouraged to identify and update vulnerable systems immediately. Many of these unpatched devices could be unmonitored RDP appliances placed by suppliers and other third-parties to occasionally manage customer systems. BlueKeep can be exploited without leaving obvious traces, customers should also thoroughly inspect systems that might already be infected or compromised."

https://www.microsoft.com/security/blog/2019/11/07/the-new-cve-2019-0708-rdp-exploit-attacks-explained/

Now here's an interesting tidbit: We've been hearing a lot about how the attempted exploitation of BlueKeep has been crashing systems. Recall that last week Kevin Beaumont first noticed the BlueKeep exploitation because the servers of his honeypot farm were crashing. Guess why?...

BlueKeep and BSODs
Okay... so, at this time the only public proof-of-concept exploit code for the BlueKeep RDP vulnerability is a module for the Metasploit penetration testing framework. That module was, in turn, assembled from proof-of-concept code donated this past summer by RiskSense's security researcher Sean Dillon. Sean tweets as @zerosum0x0.

As we know, although the exploit code works it has had the tendency to crash the systems it's targeting rather than delivering a remote shell to the attacker. As I just mentioned, it was when 10 of Kevin Beaumont's 11 RDP honeypots crashed that the world was first alerted to a BlueKeep-based campaign.

We often quote Bruce Schneier's astute observation that attacks never get worse they only ever get better. And, inevitably, recent events with BlueKeep follow that path. This past weekend the BlueKeep metasploit module was fixed to entirely eliminate the source of the BlueKeep BSOD crashes.... And you're not going to believe what it was that was originally tripping up BlueKeep: It was Microsoft's patch for the Meltdown flaws in the Intel processors.

Quoting from an interview Sean Dillon gave over the weekend, he said: "From looking at screenshots of the analysis Marcus 'MalwareTech' Hutchins did, we know code execution was achieved but that the honeypots were crashing because the exploit did not support kernels with the Meltdown patch installed."

Sean's original exploit incorporated a "KVA Shadow mitigation bypass" (Those with a good memory may recall that "KVA Shadow" is the official name for the Meltdown patch.) But another researcher pointed out that there was a way to bypass the need for the Syscall hook that was causing the trouble. So Sean's updated exploit, now posted to Github and incorporated into the Metasploit framework changes the exploit routine early in a BlueKeep attack, so a Meltdown patch bypass is no longer needed, and the early provisional BlueKeep raises to fully a weaponized state: 100% reliability and no more BSODs.

https://github.com/rapid7/metasploit-framework/pull/12553

BlueKeep and Marcus Hutchins

Last Friday, our friend Marcus Hutchins (aka Malwaretech) who will always have the distinction of having fortuitously stopped incredibly destructive WannaCry Internet worm before it could do as much damage as it would have, weighed in on the issue of a BlueKeep-based work. He posted the following Tweet thread, which I'm going share in its entirely since he raises a number of very good points, and because he knows what he's talking about:

https://twitter.com/MalwareTechBlog/status/1192926816370970625

When the news broke about BlueKeep exploitation in the wild, most of the reactions were basically "it's not a worm, so it doesn't matter". I decided I'd do a thread on why that's wrong, and why a worm isn't even a worst case scenario.

THREAD:

There are 2 main purposes of a worm (self propagation).

1. dealing with cases when there are too many vulnerable systems to reliably infect with just scanning alone.
2. dealing with a large disparity between the number of external and internal facing vulnerable systems.

The WannaCry worm served both of these purposes. Firstly, there were too many vulnerable systems to infect with just a scanning servers. Secondly, if a network had SMB exposed, then the chances that every single device on that network was vulnerable were very high.

BlueKeep is different. Not only is the number of externally facing vulnerable machines low enough to infect with a couple servers. But also, RDP is only enabled by default on Windows Server operating systems.

Because Windows clients don't expose RDP by default, unlike SMB, a BlueKeep worm wouldn't be able to pivot to systems within a network like WannaCry did. Furthermore, I'd guess it's fairly likely that if one of the network's RDP servers is exposed to the internet, they all are.

For all these reasons, a BlueKeep worm would not be hugely effective and not at all like WannaCry. They might infect marginally (not exponentially) more systems, but the downsides are huge.

A worm would not only attract a lot of attention, but be technically challenging due to the limitations of BlueKeep. The exploit is both unstable and non-generic (the attacker would need to somehow fingerprint the OS and exploit accordingly).

Building a worm in a way that doesn't just repeatedly crash every BlueKeep vulnerable system would be challenging, and by no means worth the reward. I'm not really worried about a worm, what I'm worried about is something that could be already happening.

Most BlueKeep vulnerable devices are servers. Generally speaking, Windows servers have the ability to control devices on the network. Either they're domain admin, have network management tools installed, or share the same local admin credentials with the rest of the network.

By compromising a network server, it is almost always extremely easy to use automated tooling to pivot internally (Ex: have the server drop ransomware to every system on the network).

The real risk with BlueKeep is not a worm. A worm is pointless and noisy. Once an attacker is on the network, they can do far more damage with standard automated tools than they could ever do with BlueKeep.

Remember all those news stories about entire networks being ransomwared? That starts with a single system being hacked. Not even a server, a normal, non admin, client system.

Attackers don't needs worms, it was just convenient in the case of WannaCry/EternalBlue.

People need to stop worrying about worms and start worrying about basic network security. Firewall your servers off from the internet, learn about credential hygiene. Occasionally worms happen, but every day there are entire networks compromised using only standard tools.

I obviously think he’s exactly right.

The other headline from last week was about DNS-over-HTTPS and the leaked COMCAST presentation to US congressional lawmakers.

Mozilla on DoH -vs- COMNCAST

I was very glad to see Mozilla take the leaked slide deck and push back hard with their own clarify and far more accurate letter to Congress. Mozilla's letter was signed by Marshall Erwin, the Senior Director of Trust and Security for Mozilla Corporation.

https://blog.mozilla.org/wp-content/uploads/2019/11/Mozilla-Asks-U.S.-House-to-Examine-ISPs-privacy-practices.pdf

I’ll share the beginning of Marshall’s letter to Congress so that everyone can get a sense for it:

Dear Chairs and Ranking Members:

We are writing to express our concern about the privacy and security practices of internet service providers (ISPs), particularly as they relate to the domain name services (DNS) provided to American consumers. Our recent experience in rolling out DNSoverHTTPs (DoH) - an important privacy and security protection for consumers has raised questions about how ISPscollect and use sensitive user data in their gatekeeper role over internet usage.

With this in mind, a congressional examination of ISP practices may uncover valuable insights, educate the public, and help guide continuing efforts to draft consumer privacy legislation. During the last two years, Mozilla, inpartnership with other industry stakeholders, has worked to develop, standardize, and deploy DoH, a critical security improvement to the underlying architecture of the internet. A complementary effort to our work to fight ubiquitous web tracking, DoH will make it harder to spy on or tamper with users' browsing activity and will protect users from DNS providers - including ISPs -that can monetize personal data. We believe that such proactive measures have become necessary to protect users in light of the extensive record of ISP abuse of personal data, including the following incidents:

- Providers sold the real-time location data of their mobile broadband customers to third parties without user knowledge or meaningful consent. In one particular case, an intermediary was found to be selling particularly sensitive GPSdata, which can pinpoint the location of users within a building, for over five years.

- ISPs have repeatedly manipulated DNS to serve advertisements to consumers. Comcast has previously injected ads to users connected to its public wi-fi hotspots, potentially creating new security vulnerabilities in websites. And last year, CenturyLink injected ads for its paid filtering software and disabled the internet access of its users until they acknowledged the offer.

- Verizon tracked the internet activity of over 100 million users without their consent through “supercookies” that could not be deleted or circumvented. This allowed Verizon to closely monitor the sites that users visited and catalogue their interests without their knowledge.

- AT&T operated a program that required users to pay an extra$29 per month to optout of the collection and monetization of their browsing history for targeted ads. While the company ended the program after public criticism, it has considered reviving it in the current deregulated environment.

Unsurprisingly, our work on DoH has prompted a campaign to forestall these privacy and security protections, as demonstrated by the recent letter to Congress from major telecommunications associations. That letter contained a number of factual inaccuracies. These have been examined in detail by others and as such will not be given an in-depth treatment here. Nonetheless, it is important to highlight the underlying premise of that letter: telecommunications associations are explicitly arguing that ISPs need to be in a position to collect and monetize users’ data. This is inconsistent with arguments made just two years earlier regarding whether privacy rules were needed to govern ISP data use.

With the 2017 Congressional repeal of the Broadband Privacy Order, a substantial gap in consumer privacy protection was created. That gap still exists today. ISPs are people's gateway to the Internet. That gateway can serve as a data collection point, providing ISPs with unique access into sensitive browsing information. That is why broadband privacy rules? would have required ISPs to get clear consent to use and share subscribers' information. However, those rules are no longer in place.

Our approach with DoH attempts to close part of this regulatory gap through technology and strong legal protections for user privacy. Mozilla's?policy establishes strict requirements for potential Firefox DNS resolvers, including requiring that data only be retained for as long as is necessary to operate the resolver service, that data only be used for the purpose of operating that service, and that partners maintain a privacy notice specifically for the resolver that publicly attests to data collection and policies. Unfortunately, ISPs often do not maintain privacy notices for their DNS services. As a result, their policies are opaque to users - it is unclear what data is being retained, how it is being used, or who it is being shared with.

DoH and the Web Browsers

Meanwhile, despite ISP Association hissy fits, DoH WILL be rolling out for ALL major browsers!

In the wake of all this DoH noise, ZDNet took it upon themselves to interview the product managers of all six major browsers (in alphabetical order): Brave, Chrome, Edge, Firefox, Opera & Vivaldi. The good news is that EVERY SINGLE ONE of these browsers either already has or soon will have support for DoH.

Brave

Tom Lowenthal, Product Manager at Brave for Privacy & Security told ZDNet: "We absolutely want to implement it. Implementing DoH is far more than just the technical work, though. We need to decide on sensible and protective defaults for the vast majority of people who don't think about their DNS configuration while making sure that we don't break things for the people and organizations who have carefully tuned their setup."

Because Brave is built on top of the Chromium open-source browser codebase, DoH support is available. However, the Brave team has yet tweaked the feature so that it works exactly the way they wish. So DoH is already there in the codebase the way the Google Chrome team designed it to work, as we've previously described.

DoH in Brave can be enabled at: brave://flags/#dns-over-https

Chrome

As we know, Google Chrome is the second browser after Firefox to add DoH support. DoH isn't yet enabled by default for everyone since Google is currently running a limited experiment with a small number of users to see how DoH fares in a real-world test. As we've noted, they take an adaptive approach, first honoring the user's existing DNS provider to see whether it supports DoH and using it it possible. If not it follows various heuristic paths.

DoH in Chrome can be enabled at: chrome://flags/#dns-over-https

Edge
A Microsoft spokesperson told ZDNet that they were supportive of DoH, but they couldn't share their exact plans. However, like Brave, the soon-to-be-released Chromium-based version of Edge already supports DoH.

DoH in Edge can be enabled at: edge://flags/#dns-over-https

Additional thoughts, tips and tricks from an Edge developer are here:
https://textslashplain.com/2019/11/06/thoughts-on-dns-over-https/

Firefox
As we know, Firefox was the first out of the gate with DoH and took some undeserved, in my opinion, arrows in its back for simply standardizing upon Cloudflare as their DoH provider. No one took the time to understand how rigorously Mozilla vetted Cloudflare. And many people who don't listen to this podcast might mistakenly believe that Cloudflare is just another CDN. But anyone who can erect a large wall of Lava Lamps and use their video images to generate true random numbers definitely stands out as an innovator. Which is what we know them to be.

DoH can be enabled in Firefox through its Settings UI.

Opera
Opera has already rolled out DoH support. The feature is disabled by default for all users but it can be enabled at any time in the stable release, and it works without users going through any additional steps. The flip side of the "no additional steps" is that Opera has followed Firefox's lead and simply routes all DoH traffic to Cloudflare's 1.1.1.1 DoH resolver.

Users of Opera's popular VPN should not, however, that the two are incompatible and the VPN must be disabled for DoH to work. On the other hand, if you're using a VPN you already have a privacy-encrypting tunnel which zips right past your ISP or service provider, so DoH is not needed in VPN mode.

DoH can be enabled in Opera at: opera://flags/opera-doh

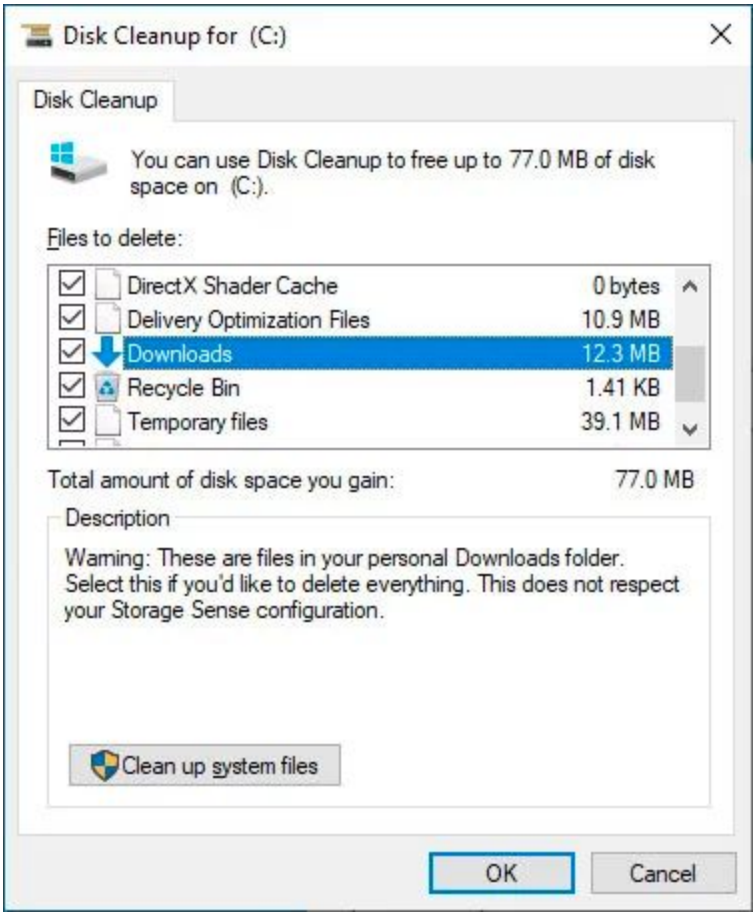
Safari
ZDNet was unable to obtain any reply from Apple about Safari but ZDNet notes that since Apple has recently been investing in user privacy-focused features, the chances are good that DoH will eventually appear in Safari.

Vivaldi
Being yet another Chromium-based browser, Vilvadi also works like Chrome.

DoH can be enabled in Vivaldi at: vivaldi://flags/#dns-over-https

Among many of COMCAST miss-assertions was their focus upon Google. Google makes a convenient soft target lately, but despite COMCAST's assertions, it's not just Google but the entire browser industry that's heading to DoH just as fast as development cycles and field trials will allow.

"Downloads" directory being removed from Windows 10 "Disk Cleanup" applet.
We previously noted here, with interest, when Microsoft decided to add the user's Downloads directory to their handy little Disk Cleanup applet. And I specifically cautioned our listeners to be careful with that one, since on the one hand many people are in the habit of forgetting their Downloads and allow it to grow without end. So I could see the logic of having that in Disk Cleanup. But in this era of multi-terabyte storage drives, many people have taken to using their Downloads directory as an archive of everything they have ever downloaded… and they assume that everything will always be there.



Anyway, Microsoft apparently discovered that too many people were selecting all of the Disk Cleanup category checkboxes without looking and were then permanently and inadvertently deleting their download archives. So that feature, which was first added into last year's il-fated and infamous October 2018 update is now being removed.

So in the future, if you want to delete stuff from your Downloads directory you'll need to do it yourself. And, really, I think that does make more sense. Disk Cleanup should be for obscure OS level stuff like the "DirectX Shader cache", "delivery optimization files","System error memory dump files" and "Windows Update Cleanup"... crap that accumulates that no one needs, like OS belly button lint.

Libarchive
The compression library included by default in Debian, Ubuntu, Gentoo, Arch Linux, FreeBSD, and NetBSD distros, contains a vulnerability that can allow hackers to execute code on user machines.

Last week, details about a major bug impacting the library went public, but this wasn't until after the Linux and FreeBSD distros had rolled out updates containing patches for the Libarchive version they had been shipping.

As we know, a decompressor is an interpreter and it's extremely difficult to find every possible flaw. And decompressors tend to be even more difficult that other interpreters because they must inherently operate with fewer constraints. So this bug, which is being tracked as CVE-2019-18408, not surprisingly allows an attacker to execute code on a user's system via a deliberately malformed archive file. Therefore, typical exploitation scenarios would involve the reception and decompressing of malicious files from attackers or local apps that use Libarchive's various components for file decompression.

The bug was originally discovered and patched back in June, but it took some time for the patch to make its way upstream to all operating systems. The vulnerability was identified by Google security researchers using two automated code testing tools: ClusterFuzz and OSS-Fuzz, and it's patched in Libarchive 3.4.0... Which is the latest release as of June 13th, 2019.

The takeaway for everyone is that now would be a good to resynchronize your OSes with the latest repository updates. You might already have v3.4.0 ... but you'll want to be sure.

There have been, and are, no known exploits in the wild, but they would likely be targeted attacks so they would tend to go unseen and unreported.

Credential Delegation

Yet another approach for solving the problem of certificate revocation within a more limited scope.

As we know, web servers prove the domains they control by providing a certificate which has been previously signed by a certificate authority whom the client knows and trusts. That certificate will contain an enumeration of one or more Internet domains and/or IP addresses and the server's previously assigned public key. During the TLS handshake, the client challenges the server with a random nonce, which the server must sign using its matching secret private key. The client uses the public key in the signed CA certificate to verify the server's signature of its message containing the nonce… and if the signature verifies the client can then be sure of one thing: that the server is, indeed, currently in possession of the private key that matches the public key in the certificate that was signed by the trusted CA.

But what if the server's private key is ever stolen?

This is such a significant and intractable problem for our industry that this podcast has spent a great deal of time through the years exploring the problem and its many solutions.

It's one thing for me to keep GRC's private key secret. I have one 1U high rack mounted server that's as well shielded from external network penetration as I've been able to make it. That physical server is in a locked rack to which only I have Level3 have access. Neither Sue nor Greg have ever had any reason to have access. That server is in a locked, heavily monitored and guarded super-secure facility, and there are no other copies of the private key anywhere in the world. So… it's about as secure as I can imagine.

But consider the situation for Facebook, Cloudflare, Google, or any other massive Internet provider. They have thousands of servers spread across the globe and hundreds of thousands of employees. Sure, many fewer employees need to have access to those servers, but certainly still a great many. We just heard that Twitter discovered some spies in their midst. This is a problem. And so, too, is keeping an **absolute secret** across thousands of networked servers around the world.

Mistakes happen. And we know that robust security is inherently always a best effort. Consequently, resiliency in the face of a breach -- which is to say recovering gracefully and quickly -- is often the best we can do.

When a server's private key escapes -- through whatever means and for whatever reason -- the best thing that can be done is to immediately revoke its authority to represent the property for which it was signed so that it cannot be used to impersonate any of those properties.

All of our long time listeners to this podcast will already know all about revocation, since its utter failure and total implementation collapse has been one of my personal crusades and hobby horses for years. Everyone will recall that I invested a great deal of time and trouble to demonstrate just how utterly broken -- non-existent, really -- Google's Chrome browser's "CRLSET" revocation handling has always been. When I deliberately created a revoked certificate to demonstrate that Chrome would honor it on every platform, they added a special case exception for that certificate. So I changed the certificate and they gave up.

Since all of these certificates eventually take themselves out of service by self-expiring, we know that it's only the "still valid" certificates that need to somehow be suppressed.

CRLs -- Certificate Revocation Lists -- were the first solution. Certificate Authorities would publish a list of all of the certificates they had issued which had been revoked but were otherwise still valid. And anyone relying upon a CA's certificate was supposed to check that list before trusting the certificate. But CA's were slow to update their lists, when they even bothered to, those lists could become quite long and slow to download, and the relying clients weren't much better about pulling and checking the lists. So even when things were being checked, there was still a significant gap in trust.

OCSP - Online Certificate Status Protocol - is a nice solution that was intended to close the large window between a certificate's revocation and client's awareness of that. But that system, too, has flaws that we've discussed in the past. Checking the status of every TLS certificate takes additional time that no one wants to spend, and if the CA's OCSP server is slow to reply, or even

worse is down, the browser is faced with a dilemma: Fail open or fail closed. If the browser fails open then any attacker can block the OCSP check to bypass that mechanism.

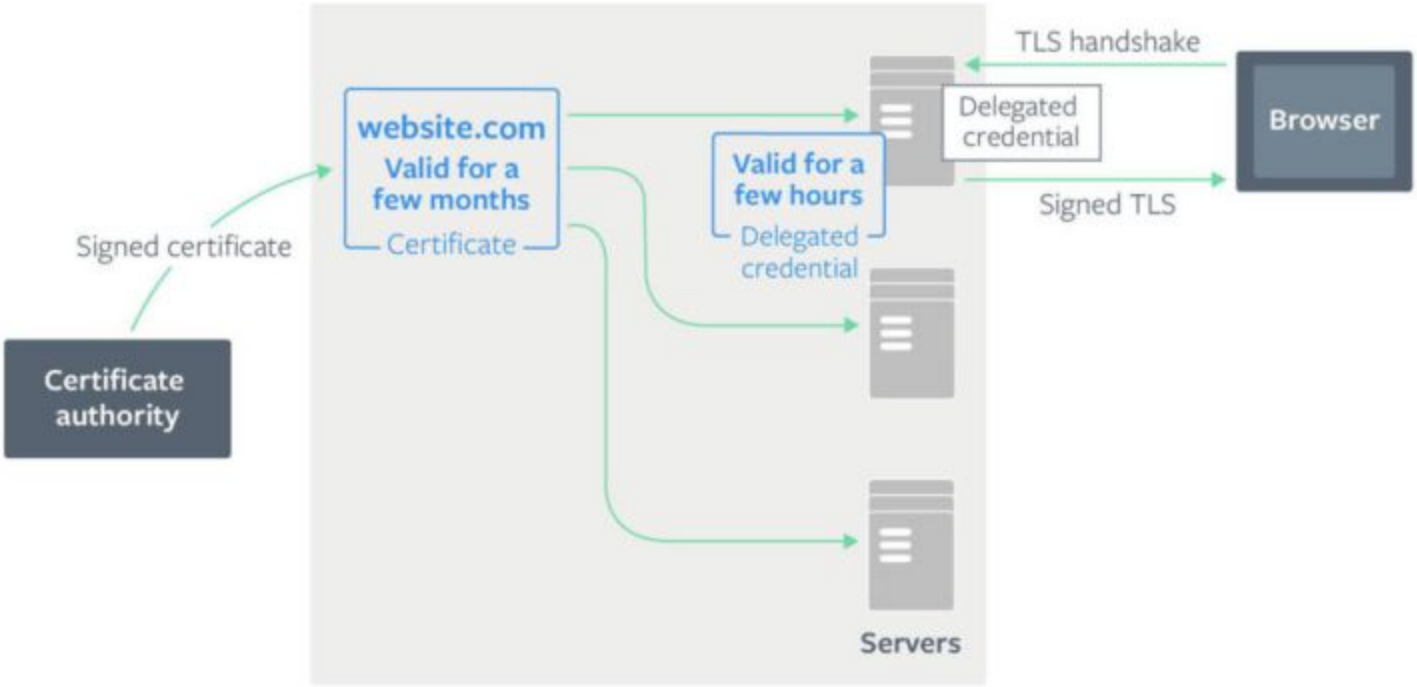
For all of these reasons, the one correct and perfect solution is OCSP Stapling. In this scenario, the web server periodically obtains a recently updated reassertion of the certificate's continuing trust and "staples" it to its original outgoing certificate. This allows the web browser to obtain a current assertion without needing to make a separate query.

But... if the web server is going to be taking the trouble to periodically obtain a fresh assertion of its certificate's validity from its certificate authority, why not just allow the server to obtain a completely new and fresh certificate with a short lifespan? And that, of course, is one of the features of the industry's latest innovation with ACME, which stands for Automated Certificate Management Environment. As we know, ACME first appeared with the LetsEncrypt movement, and they were followed by other CA's including my favorite, DigiCert.

ACME gives us a sort of workable compromise but there's a frightening tradeoff between safety and failure. LetsEncrypt's certificates are valid for 90 days. That’s a lot better than 3 years of validity, but it’s still a large open window. If a recently issued LetsEncrypt certificate is compromised, that compromised certificate will have an expiration date nearly 90 days in the future, during which a great deal of damage can be done.

And, if we were to automate the issuance of VERY short duration certs -- like hours instead of months -- then we face the possibility of the CA suffering a DDoS attack or other network outage during which and all of our short-duration certs would expire without being replaced, which would lead to a total loss of service. So that's no good either.

So the solution to this problem is a new in-the-works system formally known as "Delegated Credentials for TLS"



https://tools.ietf.org/pdf/draft-ietf-tls-subcerts-05.pdf

It's been quietly in the works for the past three years by Facebook, Cloudflare, Mozilla and Cisco... and it has just surfaced.

Because it requires support from our web browsing clients, it's going to take some time to become fully deployed, but it offers a new and very useful tool that would be practical when dealing with these high-risk scenarios involving many thousands of servers that all need to keep a secret.

It's an extension to TLS v1.3. Once our browsers have been updated to be "Delegated Credential" aware, they will send a flag in their initial "Client Hello" message indicating their support of this "Delegated Credential" extension.

We already have the well-established concept of a certificate hierarchy. The browser's awareness of this extension allows a new and final "tier" in the hierarchy to be added: A master central credential server belonging to, for example, Facebook or Cloudflare, is able to distribute very short lived -- like, just a few hours -- "Delegated Credentials" to its thousands of actual end-point web servers.

This delegated credential consists of:

- a public key,
- the expiration date of delegated credentials (the new private key), and
- the signature of delegated credentials signed by the server's leaf certificate.

This system reuses the strength of the existing certificate hierarchy by allowing content providers to, themselves, provide the last link in the chain for the issuance of short-duration TLS certificates.

These delegated credentials have no revocation mechanism of any kind, so they are hard limited to a maximum validity duration of 7 days. Clients MUST refuse any delegated credential with a life longer than 7 days.

Facebook has already implemented delegated credentials in Fizz, its open-source implementation of TLS 1.3. So we can give it a try with Firefox, which also has its support running.

Go to "about:config" and search for "delegated". Set it to TRUE and goto:

https://www.fbdelegatedcredentials.com/