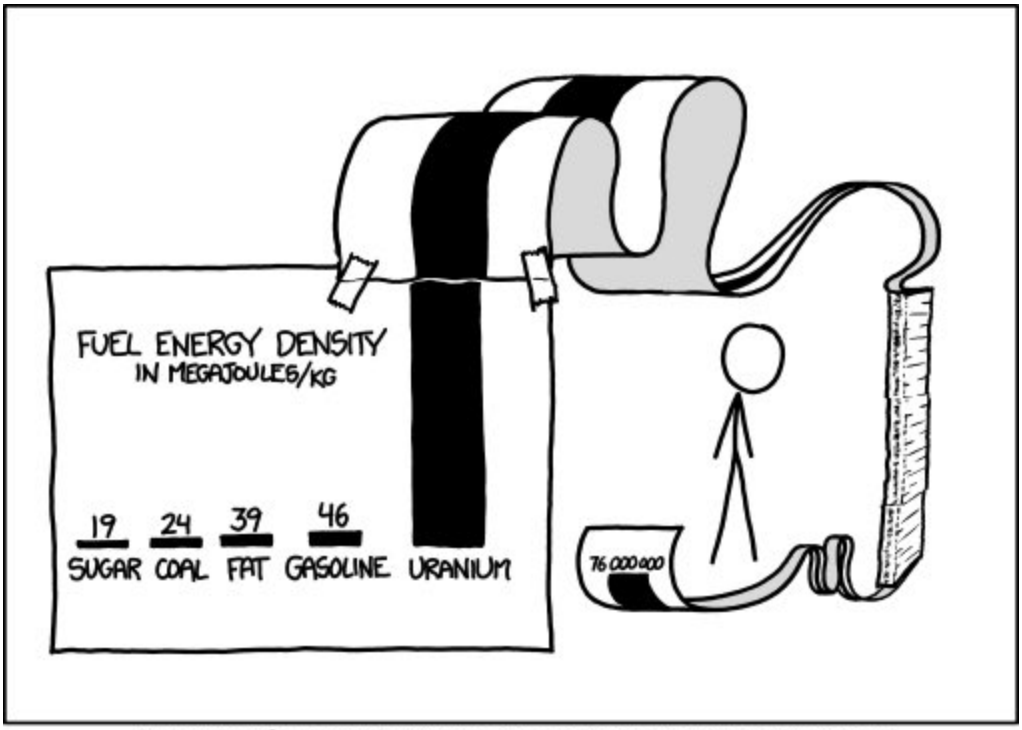
Security Now! #884 - 08-16-22

TLS Private Key Leakage

(The final podcast of our 17th year. Next week we start into year 18!)

This week on Security Now!

This week we look back at last week's Patch Tuesday to learn how much better Microsoft various products are as a result. We look at Facebook's announced intention to creep further toward end-to-end encryption in Messenger, and at the puzzling result of a recent scan of the Internet for completely exposed VNC servers. I want to take a few minutes to talk about the importance of planning ahead for a domain name's future, share my tip for a terrific website cloning tool, and a few more updates. Then, after sharing some feedback from our ever-attentive listeners, we're going to address the question: *"Can a remote server's TLS private key be derived simply by monitoring a sufficient number of its connections?"* What?! We all know that everything has been designed so that's not possible. But edge cases turn out to be a surprising problem and the details of this research are quite interesting.



Security News

Patch Flashback Tuesday

After last Tuesday's monster Microsoft patch event, so far it appears that nothing new was badly broken. But it was still quite a month in which 121 new patches were delivered. In fact, so many things received updates that it's unclear whether anything didn't There were identified CVEs in Windows and Windows components; Azure Batch Node Agent, Real Time Operating System, Site Recovery, and Sphere; Microsoft Dynamics; Edge; Exchange Server; Office and its Components; PPTP, SSTP, Remote Access Service PPTP; Hyper-V; System Center Operations Manager; IIS; of course the Print Spooler; and Windows Defender Credential Guard.

And all of that was on top of the 17 additional CVEs patched in Edge plus another 3 patches for secure boot. So this brings this month's total CVE-patched count 141. And if you were thinking that this seems like a lot, it's nearly triple the size of last year's August release — talk about year over year inflation — and the second largest release of 2022.

Of those 121 new CVEs, 17 are Critical, 102 are Important, one has only Moderate severity and the last is low in severity. Two of the bugs are publicly known with one of those, affectionately known as DogWalk, IS, or at least was, under active attack at the time of release and presumably will be until last week's patches are widely deployed.

Not long ago we were talking about the trouble with Microsoft's custom protocol handlers and how they could be readily abused. This critical 0-day remote code execution vulnerability is another of those. And as before, Microsoft was responsibly informed of this problem and declared that it was not a security problem. I suppose it wasn't until it was, since now it is, and it's being exploited in the wild to damage Microsoft's users for which, as we know, they take no responsibility.

The trouble is known as "DogWalk" because it's another path traversal flaw — thus walking the directory hierarchy. When a targeted victim opens a specially crafted ".diagcab" archive file which contains a diagnostics configuration file, it allows an attacker to save a malicious executable file into the user's Windows Startup folder. Then, the next time the victim logs into the system after a restart that file will automatically be run. The vulnerability affects all Windows versions, starting from Windows 7 and Server Server 2008 through to the latest releases.

This problem was originally responsibly reported to Microsoft on December 22nd, 2019 by security researcher Imre Rad. Even though a case was opened the next day, six months later Microsoft declined to fix the issue. Microsoft told Rad that to make use of the attack an attacker would need *"to create what amounts to a virus, convince a user to download the virus, and then run it."* [Yeah, gee... like maybe some malware.] Microsoft said that "this wouldn't be considered a vulnerability. No security boundaries are being bypassed, the PoC doesn't escalate permissions in any way, or do anything the user couldn't do already," Microsoft told Rad. Wow. So, apparently because the proof of concept was only a proof of concept and not an explicitly and massively destructive bug, and since in an sense it was by design, Microsoft said not a problem. In fact, they wrote: "There are a number of file types that can execute code in such a way but aren't technically 'executables.' And a number of these are considered unsafe for users to download/receive in email, even '.diagcab' is blocked by default in Outlook on the web and other places."

Wow. So much for layered security and defense in depth. Back then, Microsoft essentially said that, yes, these sorts of files are indeed unsafe for users to download and receive. But don't worry, our eMail client, Outlook — which, as an aside, has recently been crashing so badly for many users that it has been unusable, but that's a different story we'll get to in a minute — our eMail client, Outlook, blocks those by default.

Consequently, despite Microsoft's assurances two and a half years ago, today we have CVE-2022-34713 identified as a **critical 0-day remote code execution vulnerability.** Since this WAS finally fixed this month, it would appear that Microsoft's position on allowing attackers to execute their code, uninvited, on a victim's Windows machines has since changed, and not a moment too soon, though about two and a half years too late. And many people were hurt.

This CVE leverages Windows' Support Diagnostic Tool (MSDT). And it's not the first time an MSDT bug has been exploited in the wild this year. (We previously talked about this.) This bug allows code execution when MSDT is called using the URL protocol from a calling application, typically from Microsoft Word. (Hmm... perhaps Microsoft missed that other way to invoke the very dangerous diagnostic feature.) There is an element of social engineering to this as a threat actor would need to convince a user to click a link or open a document. What?! Well, we have nothing to worry about, then. All user's are smart and they would never click on a link or open a document.

Recall that we were also just talking about files downloaded from the Internet being tagged with the "MOTW" flag? — the Mark of the Web? The idea being that this would cause Windows to refuse to proceed or to at least pop-up a warning? Well, Microsoft's Support Diagnostic Tool which fields these ".diagcab" files was designed to not check for the Mark of the Web so that its files can be opened without any warning.

0patch's founder told The Hacker News that: *"Outlook is not the only delivery vehicle: such files are cheerfully downloaded by all major browsers including Microsoft Edge by simply visiting(!) a website, and it only takes a single click (or mis-click) in the browser's downloads list to have it opened. No warning is shown in the process, in contrast to downloading and opening any other known file capable of executing an attacker's code."* What could possibly go wrong?

The infamous SMB Server Messages Block protocol is back with another remote code execution vulnerability. It's a server side bug which allows a remote, unauthenticated attacker to execute code with elevated privileges on affected SMB servers. That's not good. But there is some good news: Only Windows 11 is affected. So only the five people using Windows 11 will be at risk. But seriously, though we don't have any details about this one, it does suggest that some new functionality added for Windows 11 also introduced a new vulnerability. If code keeps changing it's NEVER going to be secure. The concern about this one, since it affects Windows 11 systems that are publicly exposing their SMB service, is that it would potentially be wormable. Disabling SMBv3 compression is a workaround for this bug, but applying the update is the best method to remediate the vulnerability. And this once again reinforces my admonition against EVER publicly exposing ANY Windows service that does not need to be offering its services to anonymous public users — which pretty much limits it to Web and eMail. SMB has always been a catastrophe. It's the reason I created ShieldsUP! when Windows was first put onto the Internet, sharing everyone's "C" drive with the world. And they just broke it again with Windows 11.

Then we have three CVEs for Exchange Server because it has multiple Elevation of Privilege Vulnerabilities. It's a bit unusual for elevation of privilege (EoP) bugs to receive a critical rating, but these qualify. These three bugs could allow one authenticated attacker to take over the mailboxes of **all** Exchange users. This attacker could then read and send emails or download attachments from **any** mailbox on the Exchange server. Administrators must also enable Extended Protection to fully address these three vulnerabilities. Another mess.
Microsoft has also been having constant problems with their Network File System code (NFS), and we have another RCE vulnerability fixed this month, making it the 4th month in a row that NFS has received a patch to close a code execution vulnerability... and with a CVSS of 9.8 it could be the most severe of the month's patches. Fortunately, there is not a huge population of people with NFS exposed. To exploit this, a remote, unauthenticated attacker would need to make a specially crafted call to a vulnerable NFS server. But this would grant the attacker elevated privilege code execution. Oddly enough, despite its CVSS of 9.8, Microsoft has this one listed as only "Important" severity. But anyone using NFS should take it VERY seriously.

And... speaking of Outlook, CVE-2022-35742 addresses that annoying Microsoft Outlook Denial of Service Vulnerability I mentioned earlier. Get a load of this one: If an Outlook user receives a specially crafted email — and a great many did recently — their Outlook executable would immediately crash and it could not be restarted. Upon restart, it would terminate again once it retrieves and processes the invalid message. And it's not even necessary for the targeted victim to open the message or to use the Reading pane. The only way to restore Outlook's operability is to access the mail account using some other eMail client (perhaps webmail) to remove the offending emails from the mailbox before restarting Outlook. But don't worry, Outlook is your first line of defense against executing any of those malicious Diagnostic Tool extension files... so you should feel completely safe.

I know I'm being tough on Microsoft. But there's a large and growing consensus that they're generally losing control of this beast they've created. I love Windows. So I sincerely hope it's just a phase they're going through as a result of what has, in retrospect, turned out to be a poor decision to outsource Windows' quality control to their early adopter consumers. Most of those people just want to screw around with Microsoft's latest crap. They're not actually interested in controlling anyone's quality. And we've noted how, in the past at least, Microsoft often ignores those outside pleas to repair problems that **have** been identified. So, this new scheme is clearly not working. Again, let's hope it's just a phase.

Last week, Paul and Mary Jo were observing that there are growing indications that consumers are not making Microsoft any money. So Microsoft appears to be more or less abandoning the consumer for the enterprise. Speaking as a consumer, that would be FAN-TASTIC! Please just ignore us, Microsoft! Leave us with Windows 10, forever. Just patch its flaws and don't change anything else. And meanwhile add all the new features you want to the enterprise. Really, just give that your entire focus and all the rest of us lowly consumers will just be quietly getting stuff done. Sounds like a great plan!

Facebook is cautiously creeping toward default E2E encryption
In a move that's sure to turn up the heat on the question of whether consumers have the fundamental right to have their interpersonal communications end-to-end encrypted, making

them truly and irreversibly private, Facebook's Sara Su, their "Director of Trust" for Facebook's Messenger app, posted last Thursday that:

https://messengernews.fb.com/2022/08/11/testing-end-to-end-encrypted-backups-and-more-on-messenger/

> *"This week, we'll begin testing default end-to-end encrypted chats between some people. If you're in the test group, some of your most frequent chats may be automatically end-to-end encrypted, which means you won't have to opt in to the feature. You'll still have access to your message history, but any new messages or calls with that person will be end-to-end encrypted. You can still report messages to us if you think they violate our policies, and we'll review them and take action as necessary."*

As we know, this is a big deal due to what I call "The Tyranny of the Default." Even though WhatsApp, based upon the beautiful Signal protocol, offers nothing other than end-to-end encryption, Facebook's native Messenger app, which offers end-to-end encryption as an option, has never been enabled by default. And even now, they're moving that forward cautiously.

Facebook is saying that this move is unrelated to the recent outrage and backlash from privacy advocates after Facebook complied with an order to reveal the content of private messages between a mother and her daughter. The order came from a Nebraska police department as part of their investigation into an abortion-related case in the wake of the U.S. Supreme Court's reversal of its previous decision — made under different Justices — in Roe vs Wade.

Despite Facebook's denial, the timing of this move is at least suggestive, but it's a welcome move and since most Facebook users just use Facebook's built-in native Messenger app, if "encrypted by default" is eventually true, it will go a long way toward protecting the privacy of many more of Facebook's 3 billion users.

VNC's inherent insecurity

The security firm "Cyble" recently scanned the Internet for instances of completely open VNC servers lacking any password protection of any kind. And they found more than 9,000 of those which they were able to freely log in to. And, as I'll get to in a moment, these were not all safe to leave open.

So first of all, for those who don't know, VNC stands for "Virtual Network Computing." It's a very old, open, platform-independent and very popular remote console access system which implements RFB — standing for Remote Frame Buffer. So, one runs a VNC server on some machine, typically any random desktop operating system, and this server allows a remote networked user with a matching client to view the machine's console and use its keyboard and mouse remotely. It's essentially Windows RDP—remote desktop protocol—but non-proprietary.

So, this firm "Cyble" scanned and found more than 9,000 machines openly exposing their access without any form of authentication protection. This begs the question, why? How is this possible? Is it the result of negligence, or error, or a misguided deliberate decision made for convenience?

Most of the exposed instances are located in China and Sweden, but the United States, Spain, and Brazil were also well represented. Cyble identified 1,555 passwordless instances in China and 1,506 in Sweden. The U.S. came in 3rd with 835, followed by Spain with 555 and Brazil with 529. And as if just having consoles exposed wasn't bad enough, Cyble found that some of these exposed VNC instances were obviously exposing industrial control systems. They wrote:

> *"During the course of the investigation, researchers were able to narrow down multiple Human Machine Interface (HMI) systems, Supervisory Control And Data Acquisition Systems (SCADA), Workstations, etc., connected via VNC and exposed over the internet."*

In one of the explored cases, the exposed VNC access led to an HMI for controlling pumps on a remote SCADA system in an unnamed manufacturing plant.

Being curious to see how often attackers are targeting VNC servers, Cyble used its tools to monitor for attacks on port 5900, which is VNC's default server port. They found that there were over six million requests over one month. Most attempts to access VNC servers originated from the Netherlands, Russia, and the United States.

And it's worth noting that this was just a scan for VNC servers requiring no authentication. Cyble's report noted that if they had raised the bar just a bit by including VNC servers using weak and easily cracked passwords, the number of exposures would have sky-rocketed. And to that end, sadly, many VNC products are old and not super-securable at best. Many only offer passwords of up to eight characters — like I said, much of this stuff is old — so they are inherently insecure.

So, once again, we face the rule of the Internet that really needs to take hold: The only servers that should be publicly exposed to the Internet are those that are intended to be accessed anonymously by the public. Any server that's not meant to be publicly exposed should be placed behind some additional layer of strong authentication protection, such as a VPN, or SSH, or an overlay network like TailScale. Yes, this will make its access a bit less convenient for its authorized users... but it will make it FAR less convenient for anonymous remote attackers.

Miscellany

The need to control domain names.

For the third time in the last couple of years, a neighbor who had heard that I knew my way around computers stopped Lorrie and me during our nightly walk to ask what she should do about her website. The common denominator in all three instances, which is why I'm mentioning this today, is that these people were not webmasters, nor were they people who were content with having Facebook host their pages. They were professionals who wanted to have their own website located at their own domain. So they got a referral from someone else, or perhaps noticed the bottom of someone else's site a mention that the site was created by "Website's R Us.". However they found a service, one way or another they used a third-party service, typically a one-person DBA firm, to function as an intermediary between a domain name registrar and a hosting provider to register the domain name of their choice, create a website and populate it with their content.

These relationships are almost always problematic because a great deal of time can be spent getting everything set up the first time. Then the site's owner invariably wants to make changes from time to time which are met with grumbling of various degrees by the webmaster who just wants this client to shut up and pay to maintain their site, as is, without actually doing any further maintenance.

Every listener here is now thinking to themselves: "Okay, Gibson, we all know this. What's the point of this wind up?" The point is, what I have seen over and over is that these relationships do not age well in the long term. And the reason is: **The Domain Name**. Without exception, none of these regular nice people, who have their own website under a domain name that they originally chose, having had it for years and having invested emotionally, intellectually and spiritually, have any actual control over what they consider to be **their** domain name. In every case their domain name was registered by someone else for them in the beginning; at a time when none of that accumulated value existed in "the name." And without an understanding of the way domain name property is managed, they never stopped to wonder what would happen in the future? Whoever it was who set all this up for them had an account with a registrar and it was there that their domain name was — and still is — registered.

This all came to mind, recently, when I was explaining to this most recent neighbor who stopped me that where domain names are concerned, the rule is not that possession is 9/10ths of the law. With domain names, possession is 100% of the law. It's a fact and a problem which has caused all manner of pain and anguish on the Internet. But so far nothing has been done. And even if something someday was done, it would involve court battles and attorneys and lots of spent money.

The time before last, the long-time webmaster of a different neighbor had died rather suddenly, being survived by his wife. Eventually, the web and eMail services were terminated and my neighbor went in search of a solution. The man's wife was nice enough, but she knew nothing about technology and my neighbor learned that all of his other clients were furious and fuming that their web domains had been allowed to expire, had been snatched up and were now hosting click-bait pages. I had to give that neighbor the bad news that she had no practical recourse.

In the most recent case, the webmaster is an 80-year old geriatric good friend of the website owner. Working through him, she has created an extensive website filled with links, artwork, plays and songs. It's a content-rich labor of love, and she's been frantic that something might happen to it. I told her that I could easily relieve some of her worry by cloning her entire website into a directory, so that it would be archived for safe keeping. The next morning I did that and she was hugely relieved.

But I explained the situation to her about the control of domain names and that if anything should ever happen to this person, in the absence of some sort of plan, she stood to lose any and all use of that domain because the way it was registered, she has no claim to it whatsoever. It's not hers, it's his. Doing a bit of "whois" poking around I discovered that he is using a service called "enom" which is for domain name resellers. "Enom" is a branch of Hover, my own chosen registrar with whom I'm very happy. So I told her that the best thing she could do would be to create her own account at Hover and then arrange with her friend to transfer the domain with all of its existing settings intact, from his Hover account to hers.

She said it would be a bit dicey because he was a friend and she didn't wish to offend him. So I told her that in the first place, he will at least understand that it is an **entirely** valid concern for anyone to have. So, if transferring the domain to her now is not an option, could he at least explain what he had in place for dealing with his own inability, for whatever reason, to continue?

I wanted to take a moment to mention this because even though we all probably manage our own domain properties, it must be that we're all aware of others who are in a similar place and it might be worth asking them about the plans their domain name providers have for their domain names upon any event that might cause their registrations to expire and be lost.

And as I was just writing this and thinking it through, it occurred to me that domain name registrations ought to have something like a "next of kin" which could be setup in advance to notify a second-tier owner of a domain's pending expiration to that it might be possible to build the handoff of what might have grown into a valuable property to someone who the domain's current registrar could pick as their backup.

And speaking of backup: Cyotek WebCopy: ***"Copy websites locally for offline browsing"***
https://www.cyotek.com/cyotek-webcopy
I mentioned making a backup of my neighbor's website. So I wanted to give a plug to the best of any website cloning tools I have found through the years. It's free but donation-ware and I dropped $10 into their pot because it's being actively maintained and improved, and it's a truly excellent windows utility. "Cyotek WebCopy"

Google's Ryan Sleevi Retweeted Jens Axboe / @axboe
Jens Axboe was quoting @cra whose self-description on Twitter says: *"Building a Better World Through Open Collaboration."* And when Jens quoted @cra he added to his retweet: *"I don't think I've ever seen anything more true posted."* So, here's the original quote of the Open Collaboration guy via Jens Axboe retweet and Ryan Sleevi's retweet retweet. And it's all been worth it. Here's the original tweet:

> *"Running a successful open source project is just Good Will Hunting in reverse, where you start out as a respected genius and end up being a janitor who gets into fights."*

SandSara Update from Ed Cano:

> *Hi everyone,*
>
> *Recently I posted a comment stating that the Sandsara App was available in the App Store. We had to make further adjustments for the android users, so it took us more time, but now, it is also available in the Google Play Store.*
>
> *You can follow the links below to download the app.*
>
> *The IOS app: https://apps.apple.com/lb/app/sandsara-app/id1550060569*

The Android app: https://play.google.com/store/apps/details?id=com.ht117.sandsaras&hl=es_MX

The app will help you to personalize your Sandsara by selecting the patterns that you like the most, adding more designs, or changing the colors of the lights.

Even when we are happy with our current development, we are working on more features for the app, so please send us any feedback so we can improve our current version.

Regarding shipping and manufacturing, the container with the 300 Sandsara has been cleared!

We are only a few days away from the first orders to receive their units. For those backers, you will receive your tracking numbers by the end of the week. And it will take another week to receive your Sandsara by your door. After clearance, the container is on its way to the US warehouse, where our partner will split everything in orders and start the individual fulfillment.

For the rest of the orders, we mentioned that all orders would be completed by late August, and we are on track to accomplish so. We expect to complete all processes in around one week. Depending on the courier, your package might take 2 to 3 weeks once we start shipping.

Please be wary of any message this week because you might receive your tracking number!

We thank you all for your support up to this point; you will be enjoying the Sandsara soon.

If you have any questions feel free to post a comment or reach us by DM, we love to receive feedback from our backers.

Regards, Ed

Closing The Loop

Alex T / @AlexT731

Hi Steve, thanks for another great show. I think the way airplanes restrict images in messaging clients is even simpler than that. I believe WhatsApp and others send image blobs to a separate endpoint for storage, and insert a reference to that blob into the message. A firewall block to the media storage endpoint would do the trick.

Thomas Apalenek / @TomApalenek

Copy-As-Path in Win 11-- Hi Steve. I use Shift-Right Click and Copy-As-Path all the time since you showed us that a few years ago. So thank you for that. I don't know how much you've played with Windows 11. [Safe to say, as little as possible.] *"Copy As Path" has finally been moved to the standard Right-click context menu. So, no Shift required any more.*

Nice!

Brad Cochran / @brc64

Hey Steve, long time listener, first time caller. I even dusted off my Twitter account to message you!

Regarding your most recent Security Now episode, it sounds like you haven't come across Attack Surface Management before (ASM). I wanted to share a bit of info about how it works. Full disclosure, I work for Palo Alto Networks and sell a competing solution called Cortex Xpanse, but because of that I can explain at a high level how ASM works.

You're right in your assumption that this is an external scan. Essentially, ASM tools scan the global public internet on a daily basis (which is possible now due to advances in computing resources) for open ports and protocols. You mentioned Shodan in the show, and that's similar. One important note here, due to the Computer Fraud and Abuse Act (CFAA) and similar laws in other countries, these do not tend to be invasive port scans (such as nmap), which would violate the law if you ran on a network you don't control without approval.

[And I'll mention that my own ShieldsUP! TCP port scanner goes to some lengths to detect an open port without ever actually completing at connection. Since ShieldsUP! emits packets under my control, if a remote system replies to my probing SYN packet with a SYN/ACK, I never follow through with the final ACK. Instead, I send an RST packet to force the half-open connection to be aborted and immediately closed.]

Microsoft's solution comes via their acquisition of RiskIQ. Presumably they're augmenting the external attack surface data collected with that tool with their own intelligence, and even tying that with data from other tools in the Defender suite (we do something similar here).

Other common players in the ASM space include: Shodan, BitSight, SecurityScorecard, Randori and Cycognito. And more are popping up all the time. This is definitely a growing cybersecurity market. I hope that helps!

SpinRite

Tom Malaher / @tmalaher

Steve: In SN883 you say "Any new hash will need to start over from scratch earning the reputation that that exact code file is trustworthy." How does this interact with creating unique downloads of SpinRite.exe for each paying customer? Aren't those in conflict?

You're 100% correct, and it's something I've considered at some length. As it has always been, SpinRite's EXEs will continue to be uniquely created for each of its users. And I now have an HSM — a Hardware Security Module from Digicert — installed on GRC's main server which will allow it to perform automated on-the-fly EV code signing, thus providing the highest integrity assurance available. So, each custom created SpinRite will be individually and validly signed, which is the best I can do. But there will be no way for each of those signed hashes to ever earn a reputation for themselves since each one will be unique. Therefore, it may become common and expected for any reputation-based anti-malware service to be warning its user that their freshly downloaded copies of SpinRite have never been seen before.

TLS Private Key Leakage

Four researchers, two each at the University of California at San Diego and the University of Colorado Boulder performed an amazing piece of work, described in their paper, titled: "Open to a fault: On the passive compromise of TLS keys via transient errors." Their work was just presented during the 31st USENIX Security Symposium held in Boston late last week.

To make sure I have everyone's attention, I'm doing to first share their paper's Abstract:

Abstract

It is well known in the cryptographic literature that the most common digital signature schemes used in practice can fail catastrophically in the presence of faults during computation. We use passive and active network measurements to analyze organically-occuring faults in billions of digital signatures generated by tens of millions of hosts. We find that a persistent rate of apparent hardware faults in unprotected implementations has resulted in compromised certificate RSA private keys for years. The faulty signatures we observed allowed us to compute private RSA keys associated with ***a top-10 Alexa site****, several browser-trusted wildcard certificates for organizations that used a popular VPN product, and a small sporadic population of other web sites and network devices. These measurements illustrate the fragility of RSA PKCS#1 v1.5 signature padding and provide insight on the risks faced by unprotected implementations on hardware at Internet scale.*

I'll get to what they did, what they found and what it means in a second. But one of the things I loved about the way they introduced the nature of the way digital systems can be fragile was to take a true-life example that we may all recall. I know I did. They wrote:

During 2009 to 2011, Toyota issued multiple vehicle recalls after hundreds of crashes had been reported relating to unintended acceleration. Initially, Toyota placed the blame on driver error, shifting floor mats, and sticky accelerator pedals. In 2013 expert witness Michael Barr testified in the Bookout v. Toyota Motor Corporation case that a single bit flip sufficed to kill a throttle monitoring task, resulting in uncontrolled acceleration. Toyota lost the case and began settling with crash victims out of court.

The exact cause of the memory corruption in Toyota vehicles was never established: it could have been buffer overflows, cosmic rays, or hardware faults. No matter the underlying cause, the existing hardware protections were insufficient, and the software was brittle in the face of hardware errors.

We don't need to look further back than the week before last when this podcast was titled "Rowhammer's Nine Lives". With DRAM, we have an unfortunate and persistent flaw where it's entirely possible for a bit of DRAM to spontaneously flip. It was during the 25th USENIX Security Symposium in 2016 that the paper "Flip Feng Shui" was delivered and in their summary they note that Flip Feng Shui relies on hardware bugs to induce bit flips in memory. They weaponized that unfortunate characteristic of weak memory operation. And we know that this can happen without a memory system being under attack. The reason DRAM can be equipped, at extra

expense, with parity checking and ECC is because DRAM memory is not perfect. Parity checking cannot correct, but it will at least catch a single bit flip. And ECC can correct such an occurrence. And all Internet packets contain checksums to detect any simple errors introduced between the time a packet is created and it is received by the other end. These measures and many others are in place because in reality, computers can make mistakes. While we would like to believe that every time we multiply the same two large prime numbers we're going to get the same result, in reality, that's true almost all of the time, but it's that "almost" that can be weaponized.

So, with that bit of background, here's how these guys describe their work, what they did and what they found. Referring back to their Toyota example they begin:

Cryptographic software engineering is—fortunately—less often considered to be a matter of life or death. Nonetheless, faults can have a similarly catastrophic impact on cryptographic systems. As prior work has shown, attacker-induced or naturally occurring bit flips can corrupt cryptographic computations, causing them to produce incorrect results, or even leak secret information or keys.

In this paper, we show that these attacks can be applied entirely passively, allowing a network adversary to derive TLS RSA private keys [simply] by observing network traffic. ***When errors occur during a server's RSA signature computation, the resulting failed handshake can give an attacker sufficient information to derive the server's long-term private key.***

[To clarify that before I continue: When a server makes a mistake during its computation of a TLS handshake's RSA signature, that handshake will fail. But due to some known vulnerabilities in the specific cryptographic operations, the way the signature is wrong leaks some information about the private side of the RSA computation.]

We demonstrate these attacks by collecting 5.8 billion TLS handshakes from two different university networks. These handshakes included 3.3 billion connections using TLS 1.2 or earlier and 2.7 billion server signatures. Over a few months, we found nearly 2,000 non-validating digital signatures from failed handshakes.

[Again, to be clear: That number should be zeee-ro, but some fault in the servers were resulting in a very low but non-zero number of failed signature computations. Nearly 2,000 out of 2.7 billion.]

Some of these failed handshakes allowed us to compute three RSA private keys associated with Baidu, a multinational technology company and top 10 Alexa site. These three keys were used to secure more than a million connections to hundreds of hosts in our dataset corresponding to dozens of Baidu's cloud-based services.

This passive attack is particularly concerning in the context of nation-state adversaries conducting mass surveillance. Unlike active attacks or remote compromise, which risk leaving evidence of tampering, passive fault analysis leaves no trace on either client or server. A network adversary only needs to observe network traffic passively and perform simple cryptographic calculations, capabilities that modern nation states are known to possess and employ for the purposes of network surveillance.

This attack is exacerbated when TLS servers and clients negotiate non-forward secure ciphers, allowing the network attacker to passively decrypt encrypted TLS payloads using the server's private key, without leaving any trace of compromise.

[We've talked about this before. Recall that I observed that expired and no longer valid web server certificates should be securely destroyed and not allowed to escape, even though they were no longer valid. The reason for that is when protocols without forward secrecy are used, the private key can later be used to fully decrypt any conversation that may have been archived for possible future use. So these guys are observing that if all of a server's traffic is stored, and if that server LATER makes some mistakes during its RSA signature computation, then ALL of the stored communications can be retroactively decrypted.]

In addition to demonstrating passive fault attacks, we also carried out active scans of TLS hosts, and performed a retrospective analysis of historical TLS scan data between 2015 and 2022 that included tens of thousands of non-validating signatures. **In total, we computed 127 private keys from these active scans.**

[By "active scan" they mean that rather than passively collecting server handshakes being initiated by random clients, they became a TLS client and actively initiated a great many connections to TLS protocol servers.]

We compare our results to active scans from a 2015 technical report by Florian Weimer of Red Hat. He appears to have been the first to observe that active scans could be used to detect or trigger these types of RSA signature faults at scale. He found that several open-source TLS libraries did not implement countermeasures against signature faults, and performed active TLS scans over a period of months that resulted in a few hundred invalid signatures that successfully compromised private keys, mostly from devices from several vendors.

Our passive analysis and recent active scans show that these problems are still present in current implementations. We were able to compute the browser-trusted private keys for a handful of user-facing web sites from sporadic faults, as well as observing dozens of certificate private keys compromised by devices. These certificates span from untrusted device default certificates to CA-signed browser-trusted wildcard certificates for entire organizations. Although all of the open source libraries we inspected have implemented countermeasures, it appears some proprietary TLS implementations are still vulnerable to this attack.

I'm not going to go deep into the math because it's not really relevant to understanding the impact of this attack. But there is some historical background that I think everyone will find interesting...

The flaw we exploit is well known in the cryptographic side channel literature, first described in a 1997 paper about an RSA key recovery attack: Almost all RSA implementations use the Chinese Remainder Theorem (CRT) optimization for modular exponentiation in RSA signing [The Chinese Remainder Theorem is a clever optimization that is commonly used], but errors that occur in one of the half-exponentiations in this algorithm can result in leaking information that can be used to derive the RSA private key. A researcher named Arjen K Lenstra had published a technical report the year before in 1996 titled "Memo on RSA signature generation in the presence of faults."

Lenstra improved the attack to require only one signature when the message is known. This attack works against any deterministic RSA signature scheme using the CRT optimization. The countermeasure to these attacks is to validate the RSA signature before sending it.

Prior to Florian Weimer's work in 2015, almost no implementations validated RSA signatures before sending.

Following the report, all of the software TLS libraries Weimer contacted implemented countermeasures, and Cavium issued a patch for their cryptographic accelerators, which appeared to be at fault for several of the vulnerable devices he discovered.

In **this** work, we find that spontaneous faults compromising RSA keys through PKCS#1 v1.5 signatures continue to be present at a low but persistent rate in both passive and active network measurements over time, despite the attention drawn to this vulnerability in 2015.

[In other words, as should hardly come as a surprise to anyone, not everyone got the memo and fixed their code. Or, since then new code was created and the wisdom of the past was lost.]

In the present era in 2022, we find that this flaw is not just present in the types of network devices that have already been observed to suffer from cryptographic implementation flaws in previous studies, but that it also affects user-facing websites and infrastructure that receive significant amounts of network traffic.

[And, of course, significant amounts of network traffic creates an increased opportunity for an extremely low-probability event to occur.]

These vulnerabilities are due to a hazardous combination of cryptographic libraries vulnerable in the face of computational errors, and the brittle nature of the RSA PKCS#1 v1.5 signature padding scheme as used in TLS 1.0 – 1.2. PKCS#1 v1.5 signature padding makes key compromise trivial in the presence of CRT faults.

Prior to TLS 1.3, handshakes take place in plaintext, providing all the information a passive network adversary needs to validate signatures on observed connections, or derive keys when errors occur. But on a more positive note, TLS 1.3 provides multiple countermeasures against these issues, including moving the key exchange earlier in the handshake in order to ensure that certificates and signatures are sent encrypted, and using RSAPSS signature padding, which prevents this type of key compromise if implemented correctly.

So, a long ago discovered, yet still present and haunting the Internet, significant weakness in the current implementation of some TLS handshake stacks. The eventual migration to TLS v1.3 will finally fix this, but TLS 1.2 support will not be disappearing for a long time. So an active adversary could insist upon negotiating under TLS 1.2 and hope to get the remote server to stumble over its RSA signature computation. If that happened, its private key could be disclosed.