Security Now! #862 - 03-15-22

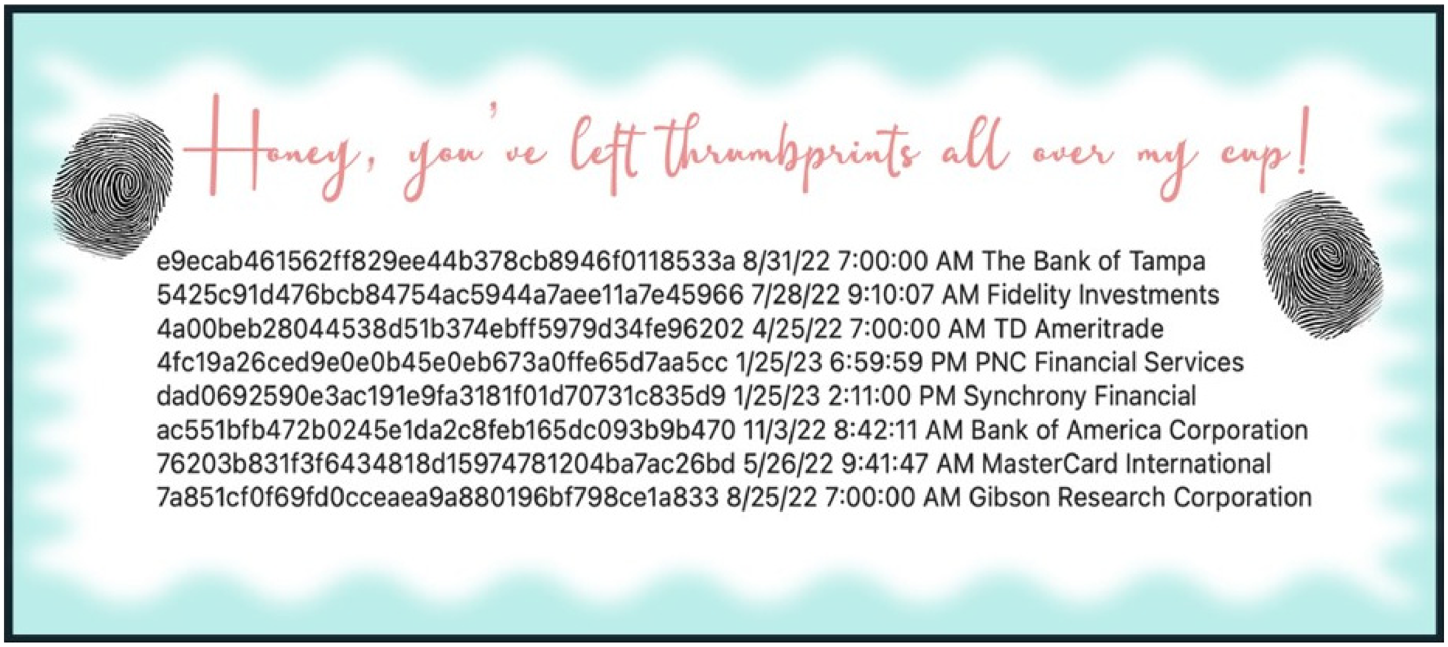
QWACs on? or QWACs off?

(Amendment to Article 45)

This week on Security Now!

This week we briefly touch on last week's Patch Tuesday for both Windows and Android, the world's two most used operating systems. We look at a recent emergency update to Firefox and the need to keep all of our systems' UEFI firmware up to date. NVIDIA suffers a huge and quite embarrassing network breach and ProtonMail handles their Russian customers correctly. The Linux kernel has seen some challenging times recently and Russia has decided to start signing website certificates. Research was just published to put some numbers to WordPress add-on's observably miserable security, and the European Union legislators, who brought us GDPR and mandatory website cookie notifications are at it again. What now?

left thumbprints all over my cup!
46f0118533a 8/31/22 7:00:00 AM The Bank of Tampa
a11a7e45966 7/28/22 9:10:07 AM Fidelity Investments
34fe96202 4/25/22 7:00:00 AM TD Ameritrade
65d7aa5cc 1/25/23 6:59:59 PM PNC Financial Services
731c835d9 1/25/23 2:11:00 PM Synchrony Financial
093b9b470 11/3/22 8:42:11 AM Bank of America Corporation
ba7ac26bd 5/26/22 9:41:47 AM MasterCard International
98ce1a833 8/25/22 7:00:00 AM Gibson Research Corporation


Honey, you've left thumbprints all over my cup!

e9ecab461562ff829ee44b378cb8946f0118533a 8/31/22 7:00:00 AM The Bank of Tampa
5425c91d476bcb84754ac5944a7aee11a7e45966 7/28/22 9:10:07 AM Fidelity Investments
4a00beb28044538d51b374ebff5979d34fe96202 4/25/22 7:00:00 AM TD Ameritrade
4fc19a26ced9e0e0b45e0eb673a0ffe65d7aa5cc 1/25/23 6:59:59 PM PNC Financial Services
dad0692590e3ac191e9fa3181f01d70731c835d9 1/25/23 2:11:00 PM Synchrony Financial
ac551bfb472b0245e1da2c8feb165dc093b9b470 11/3/22 8:42:11 AM Bank of America Corporation
76203b831f3f6434818d15974781204ba7ac26bd 5/26/22 9:41:47 AM MasterCard International
7a851cf0f69fd0cceaea9a880196bf798ce1a833 8/25/22 7:00:00 AM Gibson Research Corporation

Security News

Patch Tuesday for the Industry

Last week, the world's dominant and increasingly rudderless desktop operating system received fixes for 71 of its known flaws, three of which had been publicly disclosed and if exploited would have provided an attacker with a means for remotely executing their code. The good news is, none are known to have been exploited in the wild.

The number of remote code execution vulnerabilities led the pack, coming in at 29. Then there were 25 elevation of privilege flaws, 6 information disclosure problems, 4 denial of service mistakes (meaning that something could be made to crash), and 3 each security feature bypass and spoofing vulnerabilities.

And on top of those 71, Microsoft's Chromium-based Edge browser also separately received fixes for 21 problems that had been identified there.

Since there are public proof-of-concept exploits for some of these, as always it's worthwhile to keep systems updated.

I'm not going to spend anymore time on last Tuesday's patches because there's no much else to talk about.

Android, too

Last week, Android, the world's dominant mobile OS also received a round of fixes. The most severe of the issues fixed was a critical security vulnerability in Android's System component that could lead to remote escalation of privilege with no additional execution privileges needed — and no user interaction needed for exploitation. So, update Android, as always.

Firefox emergency update
Just over a week ago, my Firefox browser jumped to update and restart itself after Mozilla pushed out-of-cycle software update to eliminate a pair of high-impact 0-day vulnerabilities, both of which it said were being actively exploited in the wild.

They were CVE-2022-26485 and '486. Both were use-after-free flaws impacting the Extensible Stylesheet Language Transformations (XSLT) parameter processing, and the WebGPU inter-process communication (IPC) Framework.

XSLT is an XML-based language used for the conversion of XML documents into web pages or PDF docs and WebGPU is the emerging standard that is expected to supplant the current WebGL JavaScript graphics library.

So, in the case of XSLT, it was found that the removal of an XSLT parameter during processing could lead to an exploitable use-after-free flaw. As is often the case with dynamic memory management, a reference count is kept by the memory management system to free the programmer from needing to do that manually. It's a convenience that so-called “garbage collection” is done in the background without the programmer needing to worry about it. But the only way the system knows it's safe to free an object that was allocated, is if and when its reference count hits zero. It's an inherently brittle technology. It's great when everything works exactly right. But mistakes are difficult to see and hard to find. Normally it needs to be seen happening, which is why this problem keeps recurring. If you wanted to explain the trouble to a non-techie you could say that Use-After-Free bugs, which can sometimes be exploited to corrupt data and execute an attacker's code arise from a confusion over which part of the program is responsible for freeing memory.

The other user-after-free glitch was caused by an error when processing messages in the WebGPU's inter-process communication framework. A remote attacker could trick the victim into opening a specially crafted web page, trigger a use-after-free error and execute arbitrary code on the victim’s system. And, by the way, the XSLT vulnerability was exploited in the same way... Just open the wrong web page and BLAMO!

Last week you would have wanted to be at Firefox 97.0.2, but just now, when I checked, it wanted me to restart, which did and moved from 98 to 98.0.1. So, you might want to check if you're using Firefox.

HP's major UEFI firmware patch-fest
We've talked a lot about the surprisingly daunting challenge of booting a computer, while maintaining true provable security. But this really should not be so surprising. The reason computers have been such an incredible breakthrough and boon for humanity is that they're so flexible. It's a machine that follows instructions. The trouble arises because there's no way for it to know whose instructions it's following. It doesn't know or care. Its limitless vulnerability arises from its incredible flexibility.

In security we have this concept of a chain of trust and that chain is anchored by a root or trust. And this applies to booting up an operating system. If the operating system is signed by its producer, any single bit that's changed will break that signature. So, the theory is, the system's firmware simply needs to load then verify the operating system's cryptographically secure signature before it turns control over to it. Doing that will prevent anyone from modifying the operating system's code.

It's a great idea. But who's watching the watcher? On one hand, having the system's startup firmware code being firm and not hard creates a vulnerability because that means it can be altered. But on the other hand, since we appear to be unable to get it right, if it's firm it can at least be improved when problems with it are inevitably found.

Fortunately, there are people who are focused upon improving the situation. At the beginning of last month, a group of researchers from the firmware protection company Binarly discovered a raft of critical vulnerabilities in the so-called “InsydeH2O” UEFI firmware, a cross-vendor firmware used by many computer vendors including Fujitsu, Intel, AMD, Lenovo, Dell, ASUS, HP, Siemens, Microsoft, and Acer.

Binarly found 23 flaws in the InsydeH2O UEFI firmware, most of them occurring in the software's System Management Mode (SMM) which provides system-wide functions such as power management and hardware control. Since the System Management Mode privileges inherently exceed those of the OS kernel, any security issues there can have severe consequences for the vulnerable system.

The flaws found in the InsydeH2O firmware can be used to:

- Invalidate many hardware security features (SecureBoot, Intel BootGuard)
- Install persistent software that cannot be easily erased
- Create backdoors and back communications channels to steal sensitive data

As I said, there were 23 flaws found with three of them obtaining CVSS scores of 9.8. Binarly’s disclosure report explained that the root cause of the problem was found in the reference code associated with InsydeH2O's firmware framework code. In other words, as we've seen before in other contexts, all of the manufacturers derived their own firmware from Insyde's firmware SDK reference code to develop their customized UEFI firmware. This is certainly reasonable. And, frankly, I'd rather have everyone working from a common code base than each rolling their own, since getting this exactly right is crucially important. So when it's fixed for one, it's fixed for all.

Insyde Software has released firmware updates to fix all of the security vulnerabilities that were identified by Binarly, and they have published detailed bulletins to assign severity and descriptions for each flaw. But, being individually customized firmware, this must then be adopted and incorporated into the firmware of each vendor and then pushed to affected products.

So, we can expect to be seeing important firmware updates coming from many of our hardware vendors.

But, then, in addition to this, just last week HP disclosed an additional 16 high-impact UEFI firmware vulnerabilities that Binary had found which affect multiple HP models, including laptops, desktop computers, PoS systems, and edge computing nodes. These flaws allow malware to survive hard drive replacement and operating system reinstallation.

A long time ago we talked about rootkits which are able to hide in plain sight simply by hooking the operating system's directory listing API to remove any of its own files from the list. As I said at the start, software is so powerful because it's so flexible. But that flexibility is so easily turned against us.

We should all be watching for firmware updates for our hardware in the coming months. And a huge thanks to Binarly for digging into Insyde's UEFI offerings.

The NVIDIA breach

NVIDIA's networks were breached late last month. The hacker / extortion group, Lapsus$, claims to have exfiltrated 1TB of NVIDIA's data during the attack and began leaking that data online after NVIDIA refused to negotiate with them.

On February 23rd, eMails and NTLM password hashes for 71,335 NVIDIA employees were leaked on the Internet. Since NVIDIA has only 18,975 employees, and the hackers claimed to have had deep total access to NVIDIA's systems, the presumption has been that the leaked credentials included past employees as well.

Then, on February 27th, four days after the leak, the hacking group Lapsus$ claimed that it had been in NVIDIA's systems for the previous week and they made a point of stating clearly that they are not state-sponsored and they are "not into politics AT ALL."

Troy Hunt's "Have I Been Pwned" site now has the leaked data and noted that many of the hashes have been cracked and are circulating within the hacking community.

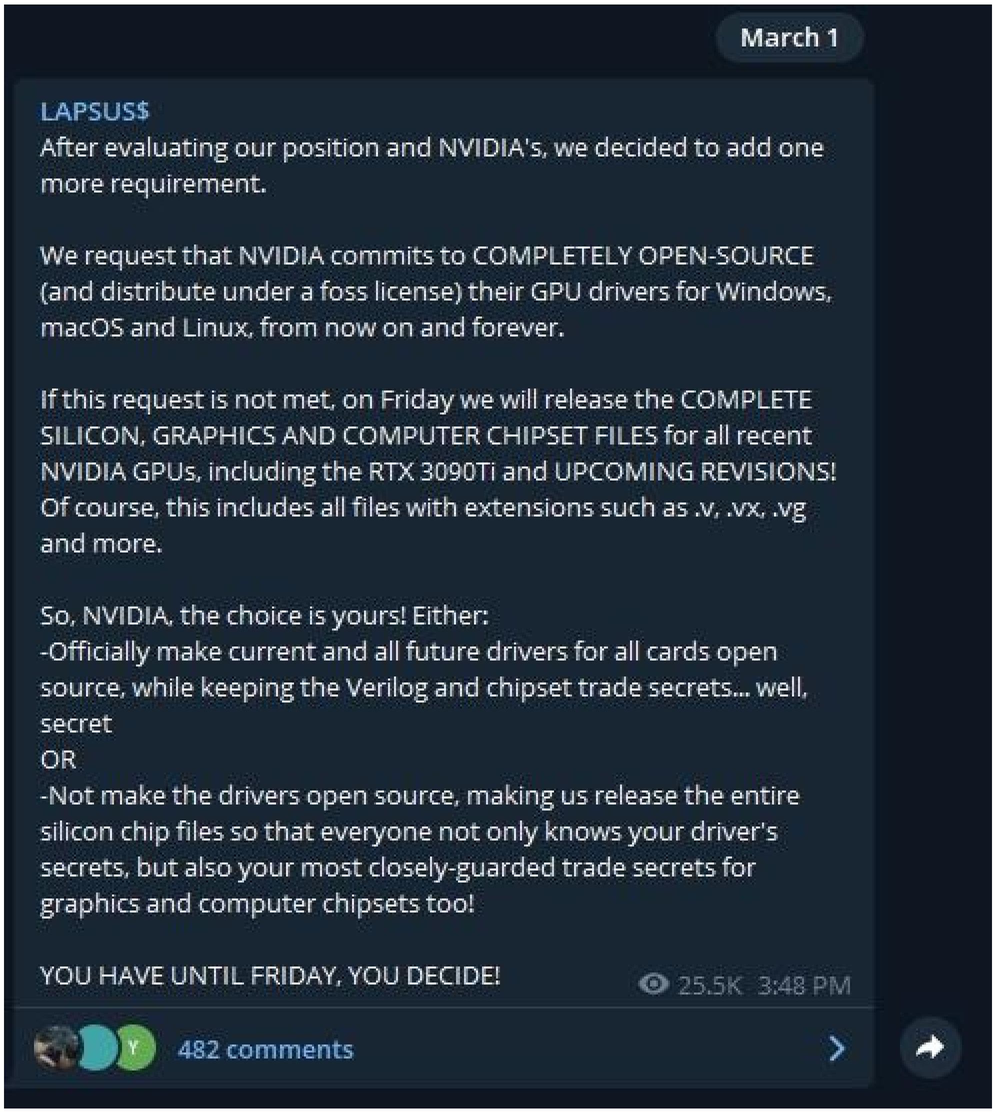
Lapsus$ also stole some expired NVIDIA driver signing certificates which were immediately put to use signing malware components to get them past antivirus and Windows' own driver signing protections. What is not clear is why Windows will load any driver signed with an expired certificate? Apparently, it will, which boggles my mind.

Code signing works differently from TLS server certificate signing. The reason is that servers and clients are always both online, by definition, and servers are always able to present a certificate that's currently valid, so that's what we force them to do.

But a driver might be signed ten years ago and the certificate with which it was signed has long since expired. Also, we want drivers to be valid even if a machine is not online. We solved this problem by changing the requirement for code signing. We require that code is signed by a certificate that is valid at the time that the code was signed. That means that we need to know when the code was signed. This information is incorporated by requiring co-signing of the certificate by a valid certificate authority which offers a timestamping service. So, at the time that code is signed, the signing computer obtains a certified timestamp from a timestamping service which it includes in the final signature. Sort of like having an online Notary. This attests to the time that the signing was done.

BleepingComputer showed the "Digital Signatures" property page of a piece of malware signed with NVIDIA's expired cert. It shows that there was no timestamp included. So, apparently, for the sake of compatibility, Microsoft doesn't enforce timestamping for some uses, such as kernel drivers, so that even stolen and expired certificates will continue to be honored. Attempts to manually verify the certificate fail because it's only the presence of a timestamp that allows the subsequent expiration of the code signing certificate to be ignored. But Windows apparently ignores it even without a timestamp.

And on March 1st, the Lapsus$ group also demanded that NVIDIA open source their drivers for Windows, Mac and Linux. Their ransom demand reads:


March 1

LAPSUS$
After evaluating our position and NVIDIA's, we decided to add one more requirement.

We request that NVIDIA commits to COMPLETELY OPEN-SOURCE (and distribute under a foss license) their GPU drivers for Windows, macOS and Linux, from now on and forever.

If this request is not met, on Friday we will release the COMPLETE SILICON, GRAPHICS AND COMPUTER CHIPSET FILES for all recent NVIDIA GPUs, including the RTX 3090Ti and UPCOMING REVISIONS! Of course, this includes all files with extensions such as .v, .vx, .vg and more.

So, NVIDIA, the choice is yours! Either:
-Officially make current and all future drivers for all cards open source, while keeping the Verilog and chipset trade secrets... well, secret
OR
-Not make the drivers open source, making us release the entire silicon chip files so that everyone not only knows your driver's secrets, but also your most closely-guarded trade secrets for graphics and computer chipsets too!

YOU HAVE UNTIL FRIDAY, YOU DECIDE!

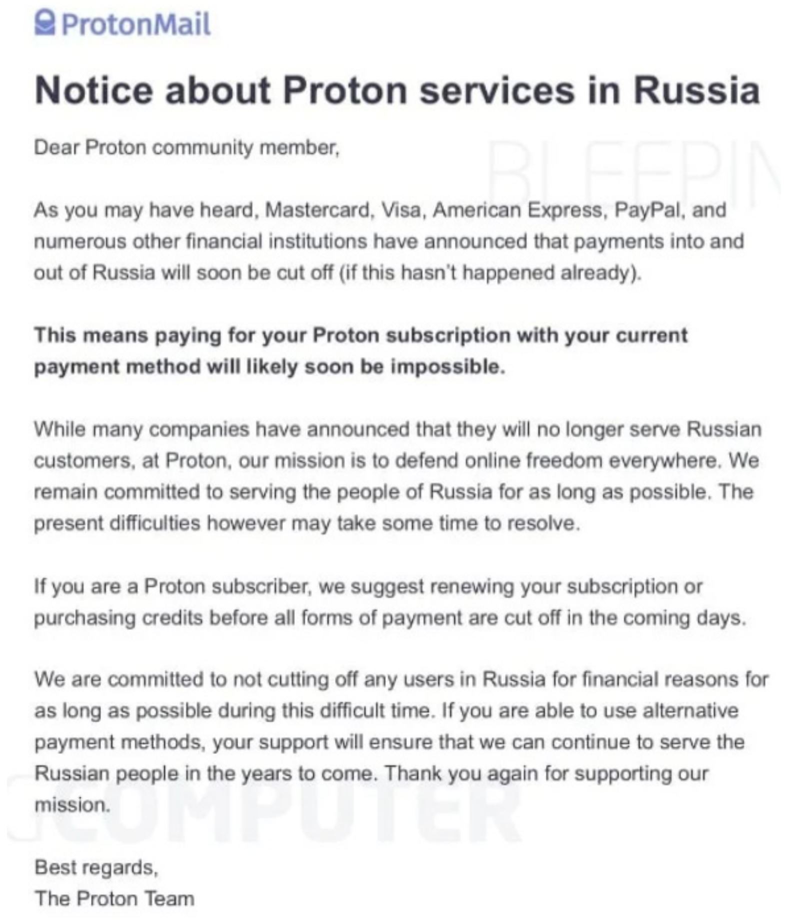
The `.v’, `.vx’ & `.vg’ file extensions are for Verilog which is the hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and manufacture of integrated circuits and it has been standardized as IEEE 1364. This means that these guys have the designs of NVIDIA’s chips. While this seems to be a bit breathtaking, it’s unclear what real value those are. No other foundry is going to make NVIDIA clone chips, they would be litigated into nonexistence. Well, maybe Russia would, except that Russia doesn’t have the process technology to make any state of the art chips. So NVIDIA’s designs would be of little use to them.

Last year, NVIDIA also came up with a technology known as Lite Hash Rate or LHR which reduces the value of its GPUs for cryptocurrency mining. Their aim was to make very powerful

Graphic Processors that would not also be very good cryptominers. This also annoyed the hackers who have said that they want the hashing rate limiters removed.

So, mostly, this is a huge embarrassment for NVIDIA, and it's a perfect example of just how much damage can be done when all of a company's proprietary value takes electronic form and can be downloaded, shared or deleted.

ProtonMail gets it right
Last week I grumbled about Namecheap's unilateral and questionable decision to dishonor their previous pre-paid commitment to their DNS domain registrants who were given one week to find another DNS provider before their domain name resolution would be summarily terminated. So I wanted to single out ProtonMail for doing the right thing with thanks to our friends at Bleeping Computer for bringing this to light.



ProtonMail

Notice about Proton services in Russia

Dear Proton community member,

As you may have heard, Mastercard, Visa, American Express, PayPal, and numerous other financial institutions have announced that payments into and out of Russia will soon be cut off (if this hasn't happened already).

This means paying for your Proton subscription with your current payment method will likely soon be impossible.

While many companies have announced that they will no longer serve Russian customers, at Proton, our mission is to defend online freedom everywhere. We remain committed to serving the people of Russia for as long as possible. The present difficulties however may take some time to resolve.

If you are a Proton subscriber, we suggest renewing your subscription or purchasing credits before all forms of payment are cut off in the coming days.

We are committed to not cutting off any users in Russia for financial reasons for as long as possible during this difficult time. If you are able to use alternative payment methods, your support will ensure that we can continue to serve the Russian people in the years to come. Thank you again for supporting our mission.

Best regards,
The Proton Team

ProtonMail provided a list of alternative payment methods, including cryptocurrency, that could be used by Russian users to renew their subscriptions. And separately, for those based in Russia, Belarus or Ukraine, another encrypted eMail provider, Tutanota, has offered to renew subscriptions free of charge for users unable to make payments in the current situation.

Linux Blues
Linux has been having a rough time of it lately. The last year has seen this ever more popular server and desktop operating system hit with revelations of multiple high profile elevation of privilege vulnerabilities. There have been problems in Linux's iSCSI subsystem, the kernel, the Extended Berkeley Packet Filter (eBPF), and, as we talked about at the time, Polkit's pkexec component. There have been two additional recent problems.

The first bears the unfortunate name “Dirty Pipe” and was discovered by a guy named Max Kellerman who discovered, mostly by accident, an important locally exploitable vulnerability which was introduced into the Linux kernel at v5.8. Because the exploitation of this flaw allows overwriting data in arbitrary read-only files it could be put to some very creative, and not generally beneficial, use, leading to privilege escalation because unprivileged processes would be able to inject code into root processes. In some ways this is reminiscent of 2016's “Dirty Cow” Linux vulnerability, though this one is even easier to exploit. And this explains, even if it doesn't forgive, this vulnerability's name.

In Max's original posting and disclosure, he explains how this all began...

It all started a year ago with a support ticket about corrupt files. A customer complained that the access logs they downloaded could not be decompressed. And indeed, there was a corrupt log file on one of the log servers; it could be decompressed, but gzip reported a CRC error. I could not explain why it was corrupt, but I assumed the nightly split process had crashed and left a corrupt file behind. I fixed the file's CRC manually, closed the ticket, and soon forgot about the problem.

Months later, this happened again and yet again. Every time, the file's contents looked correct, only the CRC at the end of the file was wrong. Now, with several corrupt files, I was able to dig deeper and found a surprising kind of corruption. A pattern emerged.

Max goes on at great length to explain how this kept nagging at him until he finally had to sit down and get to the bottom of it. After a great deal of analysis and experimentation he finally figured out how to force the file corruption bug to occur at will. That was the key. Max wrote: *“All bugs become shallow once they can be reproduced.”* So, he then tracked it down to a bug, which he located, in the Linux kernel. He checked Linux 5.10 (Debian Bullseye) which had the bug, but the bug was not present in the previous Linux 4.19 (Debian Buster). Those two releases were separated by 185,011 Git commits. But using Git's various tracking tools he was able to locate the ONE commit, made nearly two years ago on May 20th, 2020, which introduced this very subtle bug. That change refactored Linux's pipe buffer code for anonymous pipe buffers, changing the way the “mergeable” check is done for pipes... and, in doing so, introduced an extremely subtle flaw. The vulnerability has been fixed in Linux 5.10.102, 5.15.25, 5.16.1.

So that was what Max Kellerman found. Two researchers at Huawei discovered a vulnerability in Linux's “control groups” feature which allows attackers to escape containment, escalate privileges and execute arbitrary commands on a host machine.

Linux control groups, or “cgroups”, allow system admins to allocate computing resources such as memory, bandwidth, and such, among whatever processes might run on a system. These cgroups provide fine-grained control over allocating, prioritizing, denying, managing and monitoring system resources. This means that cgroups can be a powerful tool for control and security within a system.

A feature known as the release_agent file allows administrators to configure a 'release agent' program that would run upon the termination of a process in the cgroup. That meant that attackers capable of writing to the release_agent file can exploit it to gain full admin privileges. And Linux simply wasn't checking that the process setting the release_agent file had administrative privileges.

All of this amounts to a container escape, for example within a Kubernetes environment which would provide attackers with access to other users' containers in public cloud environments.

And we're not finished with Linux yet.

Yesterday, another just disclosed security flaw came to light in the Linux kernel. This one can be leveraged by a local adversary to gain elevated privileges on vulnerable systems to execute arbitrary code, escape containers, or induce a kernel panic.

It's CVE-2022-25636 with a CVSS of 7.8 and it impacts Linux kernel versions 5.4 through 5.6.10. It's the result of an out-of-bounds write (in other words, a buffer overrun) in the heap of the Kernel's netfilter component. The flaw was discovered by Nick Gregory, a researcher with Capsule8.

Red Hat explained that: "This flaw allows a local attacker with a user account on the system to gain access to out-of-bounds memory, leading to a system crash or a privilege escalation threat." In addition to Red Hat, similar alerts have been released by the maintainers of Debian, Oracle Linux, SUSE, and Ubuntu.

Linux's Netfilter is a framework provided by the Linux kernel that enables various networking-related operations such as packet filtering, NAT and Port translation. The problem results from incorrect handling of hardware offload that could be weaponized by a local attacker to cause a denial-of-service (DoS) or possibly execute arbitrary code.

The idea of hardware offloading is that there are a number of things associated with sending and receiving network packets that are very simple to do, such as calculating a packet payload's checksum, and so are a waste of the processor's time. Another example of something that can be offloaded to smart hardware is the coalescing of smaller packets into larger composites to reduce the overall packet count at the hardware interface level. So, over time, these duties have been pushed down to the hardware.

Nick Gregory, the researcher who discovered this particular problem said: "Despite being in code dealing with hardware offload, this is reachable when targeting network devices that don't have offload functionality, for example, the local virtual interface. This can be turned into kernel [return-oriented programming]/local privilege escalation without too much difficulty, as one of the values that is written out of bounds is conveniently a pointer to a net_device structure."

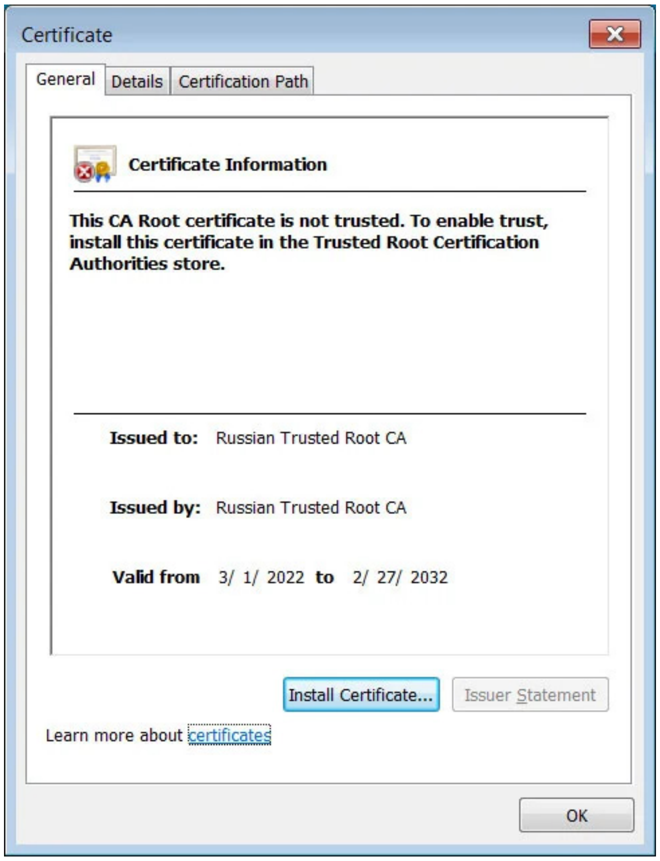
Okay. So...

Does the recent spate of flaws suggest that Linux is becoming rickety? No. It is, however, becoming increasingly complex. And we know that security and complexity are perpetually at odds with one another. For security, scrutiny is always a good thing. At this point it's the only thing that's saving us. What we've seen is that open source doesn't automagically bring more security, but in the long run it really does feel like the right solution.

Russia's New CA

Okay. So our web browsers and operating systems all collectively trust any security certificate, such as a brand new web server TLS certificate that's never been seen before, but which has been signed by a Certificate Authority that **is** trusted globally by the entire group, specifically and only because it has promised to, and demonstrated its ability to, restrict its issuance of certificates to entities whose identity it has confirmed without the guidelines setup by the CA Browser forum.

Several times during this podcast's life we've discussed instances of apparent mistakes being innocently made, flaws in certificate issuing systems being exploited, and also clear instances of deliberate abuse. So, everyone is a bit concerned over Russia's recent announcement of its creation of a new, unvetted, untested and very likely rogue Russian Certificate Authority:



Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: Russian Trusted Root CA

Issued by: Russian Trusted Root CA

Valid from 3/ 1/ 2022 to 2/ 27/ 2032

Learn more about certificates

Source: BleepingComputer

However, its creation was probably inevitable because the sanctions imposed by Western companies and governments are preventing Russian web services from renewing their expiring TLS certificates. And we all know that browsers take a very dim view of websites which offer them an expired certificate as proof of their identity.

So, if web service providers, whose certificates are expiring, are unable to purchase certificates from traditional certificate authorities... what choice do they have? Hey, no problem, just pick up a certificate issued by your local authoritarian regime.

Now there's only one problem with doing that, which is that no mainstream web browser will accept any certificate randomly signed by Russia's newly minted CA. That's just not going to happen in this lifetime. Oh, but there is an exception, Russia does have the Yandex browser, based in Russia, and it will now work with the Russian Ministry of Digital Development issued certificates. So Russian sites are instructing their visitors to please switch over to Yandex.

But, of course, there is an alternative: If someone wished to continue using Firefox or one of the several Chromium-based browsers, in theory they could do so by adding the “Russian Trusted Root CA” certificate to their browser's certificate root store. But just to be clear, that is the LAST THING that anyone outside of Russia in the west should consider doing. NOTHING would or will prevent Russian Kremlin Gremlins from creating TLS certificates for any western websites, thus allowing them to intercept, spoof and alter western website’s network traffic.

What will likely happen, and it’s probably already in the works, is that Chromium and Firefox will specifically blacklist Russia's rogue root certificate so that no one using Firefox or a Chromium browser could be hurt after mistakenly installing it into their browsers.

So, for those using the Yandex browser within Russia, it's a workable solution for providing continuity of services. But there's no conceivable way that any of our mainstream browsers are going to trust any certificates signed by Russia's new Certificate Authority.

The state of WordPress security

You may have thought that you were going to escape this week without having to listen to me harp on WordPress. No such luck. But I do promise to keep this brief. And so as not to introduce any of my own bias, I’m going to simply quote verbatim from just the top of BleepingComputer’s coverage of a recently released report on WordPress security. BleepingComputer wrote:

> Patchstack, a leader in WordPress security and threat intelligence, has released a whitepaper to present the state of WordPress security in 2021, and the report paints a dire picture. More specifically, 2021 has seen a growth of 150% in the reported vulnerabilities compared to the previous year, while 29% of the critical flaws in WordPress plugins never received a security update. This is alarming considering that WordPress is the world's most popular content management system, used in 43.2% of all websites out there. Of all the reported flaws in 2021, only 0.58% were in WordPress core, with the rest being on themes and plugins for the platform, coming from various sources and different developers. Notably, 91.38% of these flaws are found in free plugins, whereas paid/premium WordPress add-ons only accounted for 8.62% of the total, reflecting better code vetting and testing procedures.

Sci-Fi

The Bobiverse? (I’m liking #4). “The Adam Project” on Netflix   &   Where was that program?”

QWACs on? or QWACs off?

QWAC stands for *"Qualified Website Authentication Certificates"* which is something the European Union has proposed to adopt in an amendment to Article 45 of their Framework for Digital Identity. In this case the Digital Identity of websites, not people. What makes this way more interesting than watching paint dry — or watching the slowly grinding wheels of bureaucracy — is that the EU has been attempting to make this happen for the last six years, since 2016, and they still haven't given up. And after the most recent proposal by the EU, a group of 38 well-informed security and IT academics and professionals, including the EFF, cosigned an open letter to EU regulators warning that the enactment of this proposal could expose Internet users to cybercrime.

Their open letter is titled: *"Global website security ecosystem at risk from EU Digital Identity framework's new website authentication provisions."* The group starts off by saying: *"Dear Honourable Member of the European Parliament, Dear Member of TELE Working Party, We the undersigned are cybersecurity researchers, advocates, and practitioners. We write to you, in our individual capacities, to raise grave concerns regarding certain provisions of the legislative proposal for a European Digital Identity framework (the 'eIDAS revision'), and their impact on security on the web."*

The EU doesn't appear to be backing down about what they want. And we've seen what a mess can be created when something like the GDPR and its cookie regulations are imposed upon the world. So I was very interested to understand what the EU is up to now. They've intruded into our lives by making us agree to every website's cookie usage. So what might be coming next?

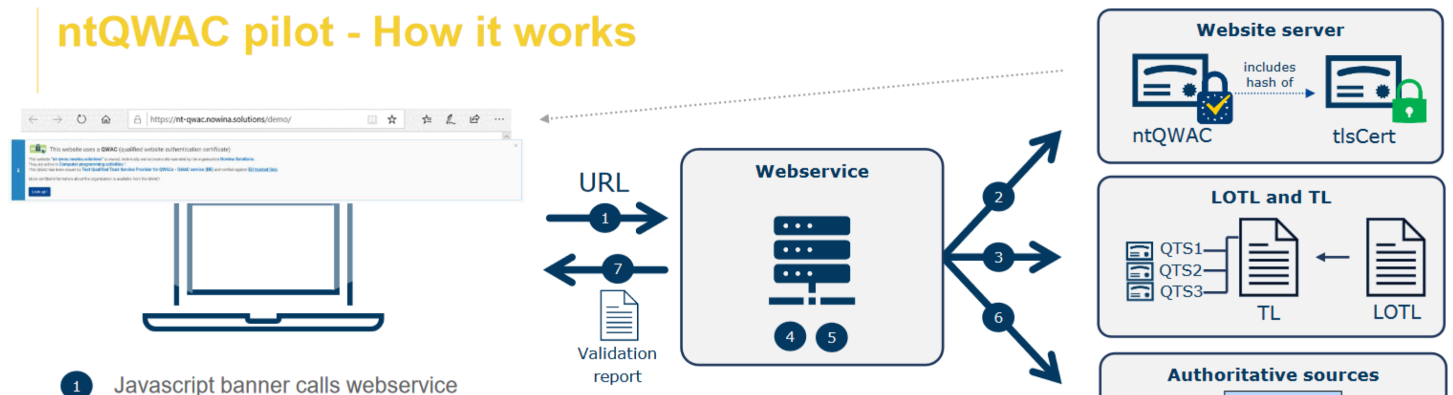
I've spent hours working to wrap my head around this. I've read the arguments being made by both sides and all of the references I could find. I did find a very strong and beautifully written explanation of the fundamental problem with what the EU wants to do, which I believe was written last July by Google's Ryan Sleevi. At least, he had a tweet which pointed to it for more information. Unfortunately, it was written last July, and the EU still doesn't appear to get it.

What the EU wants is something more than TLS certificates. They want to have their own website authentication. The ability to bind their own array of Governmental validation and authentication assertions to websites. When an EU citizen is browsing the web and goes to a website, they want the user's browser to display a top-of-page banner stating that: *"This website uses a QWAC (Qualified Website Authentication Certificate)"* and then a list of a few privacy and security assertions, in a banner which will have three tabs: "Summary", "ntQWAC Validation", and "TLS validation". There will also be an ***"I got it"*** button to dismiss the banner notification.

How is this done? The website would serve some JavaScript. (Perhaps the same JavaScript that now makes us all agree to be cookied.) This JavaScript makes a call to an external webservice which is tasked with producing a report which will be returned to the user for display.

The web service **(2)** retrieves another blob, known as the ntQWAC from a well-known default path on the website the user is visiting. That blob contains a number of government provided security and other assertions about the site and a hash of the website's TLS certificate.

ntQWAC pilot - How it works



1. Javascript banner calls webservice
2. Webservice (WS):
   downloads ntQWAC from "https://…/.well-known/eidas/"
   downloads tlsCert (e.g. EV, DC, OV cert)
3. WS validates ntQWAC against MS trusted list according to ETSI TS 119 615
4. WS verifies ntQWAC and tlsCert cryptographic relationship
5. WS processes ntQWAC attributes to produce structured ntQWAC validation report
6. WS performs additional checks against authoritative sources (e.g. VAT against VIES, LEI against GLEIF)
7. Javascript receives back the validation report
8. ntQWAC validation report is displayed into three tabs (Summary; ntQWAC Validation; TLS validation)

Webservice could be run by a TSP and the ntQWAC validation report signed/sealed

10

Since this is just a blob on a website, the web service needs to have the blob validated. So it queries an online validator **(3)** to verify the authenticity of the ntQWAC blob. It then hashes the orginal website TLS certificate and **(4)** compares it to the hash it expects. That must be to prevent certificate forgery, such as we were mentioning with the BGP routing hack the other day. So now it knows that it's associated with the proper TLS certificate. Next, it processes the ntQWAC attributes to **(5)** produce a structured ntQWAC validation report and it **(6)** performs online on-the-fly validation against additional authoritative sources.

And when all that's done the webservice **(7)** returns the validation report to the waiting Javascript which then **(8)** presents the banner to the EU citizen... who will have become quickly trained to click the "Okay, I got it" button just as he or she needs to click "Okay to Cookies" now wherever they go.

So, what do we get in return for all this industry? We get what the EU apparently wants, which is its own governmentally-controlled facility for determining, controlling and displaying various privacy and security assertions about individual websites. The EU essentially wants to add a layer of their own input — much as they have with cookies — into the browsing experience of all of their citizens.

There are a number of problems with this. But as I said, these problems have been patiently, carefully, articulately and repeatedly laid out and explained... so far, to no avail.

One biggie should stand out to everyone right off the bat: The web has been deliberately designed to be a two-party system. The user with their browser and whatever web site they are visiting. While it's true that in today's world this pure two-party system immediately collapses due to a website's voluntary and deliberate inclusion of 3rd parties, that's a site's choice to make, and we have been carefully chronicling all web browsers moving toward ever greater and stronger isolation of any and all 3rd parties. But the EU's proposal wires the 3rd party ntQWAC web service into every page's display, allowing for tracking and profiling.

The larger problem that occurs to me is what prevents a malicious website from simply displaying the same happy news ntQWAC banner, and giving its intended victim the "All Clear!"? If the system is that easily spoofed then what's the point?

The EFF, in their never boring and always over-the-top sky-is-falling style, issued a Press Release on March 3rd:

SAN FRANCISCO—Electronic Frontier Foundation (EFF) technologists, along with 36 of the world's top cybersecurity experts, today urged European lawmakers to reject proposed changes to European Union (EU) regulations for securing electronic payments and other online transactions that will dramatically weaken web security and expose internet users to increased risk of attacks by cybercriminals.

The ill-conceived proposed amendment to Article 45 in the EU's Digital Identity Framework (eIDAS) requires popular browsers like Firefox, Google, and Safari to accept flawed website certificates that bypass the rigorous security standards built into today's browsers to ensure user data isn't intercepted and stolen by criminals. Website certificates help ensure that, when you use a credit card to buy something online, your payment information is going to the right website and not to cybercriminals who have created fake websites that impersonate real websites.

In a letter today to members of the European Parliament, EFF Director of Engineering Alexis Hancock, EFF Director of Technology Projects Jon Callas, and cybersecurity experts from Belgium, Canada, France, Germany, Taiwan, the UK and the U.S. said requiring browsers to accept Qualified Website Authentication Certificates (QWACs), a specific EU form of website certificate that never gained traction because of implementation flaws, would put the entire website security ecosystem at risk by requiring browsers to trust third parties designated by the government without any security assurances.

The experts urged EU lawmakers to amend the revised Article 45.2 to "ensure that browsers can continue to undertake their crucial security work to protect individuals from cybercrime on the web." Insecure third parties can have a devastating effect on online privacy and security by opening the door to malware attacks, stolen personal and financial information, and other acts of cybercrime.

"While we understand that the intent of these provisions is to improve authentication on the web, they would in practice have the opposite effect of dramatically weakening web security," 38 cybersecurity researchers, advocates, and practitioners said in the letter. "At a time when two-thirds of Europeans are concerned about being a victim of online identity theft and over one-third believe they are not able to sufficiently protect themselves against cybercrime, weakening the website security ecosystem is an untenable risk."

Website authentication is a cornerstone of security online and the basis for e-commerce and e-government. Internet users become at risk for cybercrime when certificate authorities are not rigorously vetted to ensure their certificates can be trusted. Signatories to the letter are holding a technical workshop today to discuss implications of the proposed amendment.

If you've been following along, you'll have noticed that what the EFF wrote sounds like they're describing an entirely different system. The reason is, there are actually several proposed ways this system could work and the one the EFF is steamed about is the worst one of them. If we can call the mess I described a two-certificate solution, there is a proposed single-certificate solution.

In that solution, various governments would each have their own root certificates added to our operating systems and browser root stores and these governments would be able to issue TLS website certificates containing all of the additional authentication information they desire. This is the point in the podcast where, in unison, we all ask... *"What could possibly go wrong?"*

The Internet Society has published an Internet Impact Brief titled: *"Mandated Browser Root Certificates in the European Union's eIDAS Regulation on the Internet"* Skipping past a bunch of the intro, they write:

The European Commission is currently evaluating this regulatory framework. It organized a public consultation in 2020 to collect feedback on drivers and barriers to the development and uptake of trust services and eID in Europe. On this basis, the Commission has proposed an amendment to the 2014 regulation to try to stimulate adoption of the framework and the availability of secure eID schemes across the internal market.

Among the changes in the new proposal are provisions around QWACs that could have the effect of forcing browsers to include TSPs (i.e. CAs) authorized by national governments into their root stores, without guaranteeing security parity with current best practices. In other words, it would require browsers to add these CAs to the list of trusted middlemen – even if they do not meet industry standards or are recognized as a trusted party.

Although the text of the proposal may seem uncontentious, the high-level goals have unstated technical consequences which we believe are unpalatable regardless of which option is put into practice (For more information about our interpretation of the proposed revisions, Annex A).

Annex A:

There appear to be three technical options, each with their own risks:

1. Cryptographically bind QWACs to TLS certificates. This violates the principle of technology neutrality, because it means that QWACs will only work if used in conjunction with a TLS session. However, as eIDAS is not in a position to require TLS, in fact its own "technology neutrality" requirements prevent this option from being implemented.

2. Have separate protocols for (i) TLS certificate validation and session establishment, and (ii) QWAC validation and user interface. This option introduces new threats to users' security and privacy, in the form of opportunities for machine-in-the-middle (MitM) attacks on the separate QWAC validation protocol, disclosure of users' browsing behaviour to third parties, and an over-all increase in the attack surface through added complexity.

3. Add eIDAS-approved Trust Service Providers (TSPs – the issuers of QWACS) to the trusted root list. This represents a security risk because it violates the strict vetting rules for inclusion in the list of trusted CAs, and opens up the dangerous possibility of a government mandating inclusion of a CA over which it exercises inappropriate control (the Kazakhstan- or Mauritius-style "government root" attack).

ETSI’s core assumption is that browsers will validate (“recognise”) a QWAC by binding it to a TLS certificate for the domain it is intended to represent (Option 3). Since TLS certificates themselves do not provide the level of authenticity eIDAS wants to achieve, this means that QWACs must be issued according to a separate set of processes and criteria for eIDAS-compliant Trust Service Providers.

At this point it’s very unclear what’s going to happen.