Security Now! #741 - 11-19-19

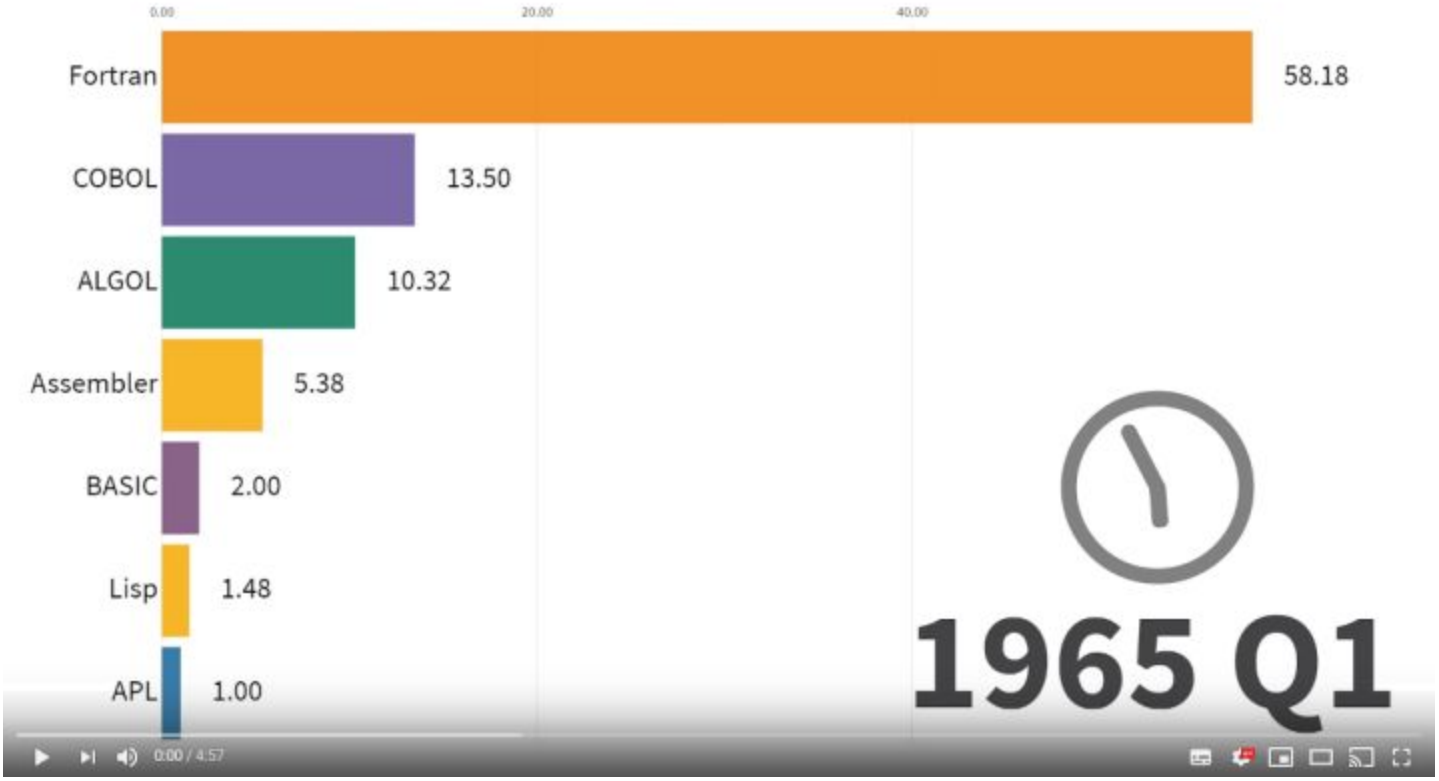
TPM-FAIL

This week on Security Now!

This week we look back a November's patch Tuesday while we count down to the impending end of patches for Windows 7 and Server 2008. We check in with CheckM8 and CheckRain as the iOS BootROM exploit continues to mature. We look at GitHub's announcement launch of "GitHub Security Lab" to bring bounties and much stronger security focus to the open source community. We discuss a recent court ruling regarding US border entry device searches. We cover yet another bad WhatsApp remote code execution vulnerability. We examine the impact of version 2 of ZombieLoad, the formation of the ByteCode Alliance, and a bit of media miscellany. Then we examine the impact of two Trusted Platform Module (TPM) failings: one which allows local key extraction and a second that can be exploited remotely over a network.

Most Popular Programming Languages 1965 - 2019

(2.6 Million Views)



https://youtu.be/Og847HVwRSI
https://grc.sc/languages

Security News

November's Patch Tuesday

November's Patch Tuesday arrived last week to resolve 73 vulnerabilities across Microsoft's software products, 13 of which were deemed to be CRITICAL, and one of those being a 0-day.

That 0-day (tracked as CVE-2019-1429) which has been found being actively exploited in the wild, is a scripting engine vulnerability in Internet Explorer reported independently by four different researchers. The vulnerability in IE allows an attacker to execute rogue code if a victim is coaxed into visiting a malicious web page, or, if they are tricked into opening a specially crafted Office document.

Microsoft wrote: "An attacker who successfully exploits the vulnerability could gain the same user rights as the current user. If the current user is logged on with administrative user rights, an attacker…could take control of an affected system."

Under an Office document attack scenario, Microsoft said an adversary might embed an ActiveX control marked "safe for initialization" in an Office document. If initialized, the malicious document could then redirect to a rogue website, booby-trapped with specially crafted content that could exploit the vulnerability.

Though IE is largely replaced by Edge under Windows 10 and probably by other 3rd-party web browsers under Windows 7 and 8, its internal rendering code exists as a core library inside all versions of Windows.

A second IE critical vulnerability is CVE-2019-1390, which identifies an issue with how the VBScript engine handles objects in memory, and raises the same concern about its exploitation in Office.

Although not known to be exploited, another reason to patch is CVE-2019-1457, which is a macro security bypass affecting the Mac version of Excel 2016 and 2019, disclosed a month ago by security company Outflank.

One oddity worth mentioning is CVE-2018-12207, which with its ID from last year (2018) looks like a mysterious 74th CVE. This turns out to be a denial of service vulnerability in Intel processors affecting guest virtual machines (VMs) which, despite its ID date, was only revealed this week by Intel. (We'll be talking about Intel security news shortly.)

But in the meantime, speaking of Intel, like Adobe, they, too, recently started synchronising their patches to coincide with Patch Tuesday. Although they don't have a impact only on Microsoft software, it's meant to be helpful.

A count of the patches reveals 9 patches for Microsoft's Hyper-V virtualization, four of which are on the list of critical flaws because they all potentially allow for Remote Code Execution (RCE).

The remaining critical flaws include 3 for the Edge browser (presumably the current Edge, not the still-unreleased new Chromium Edge browser), one affecting Exchange Server 2019, and four in various other Windows components.

A final issue Microsoft mentions is a weakness which has been discovered in the way STMicroelectronics' Trusted Platform Modules (TPMs) implement the Elliptic Curve Digital Signature Algorithm (ECDSA) in their version 2.0 hardware.

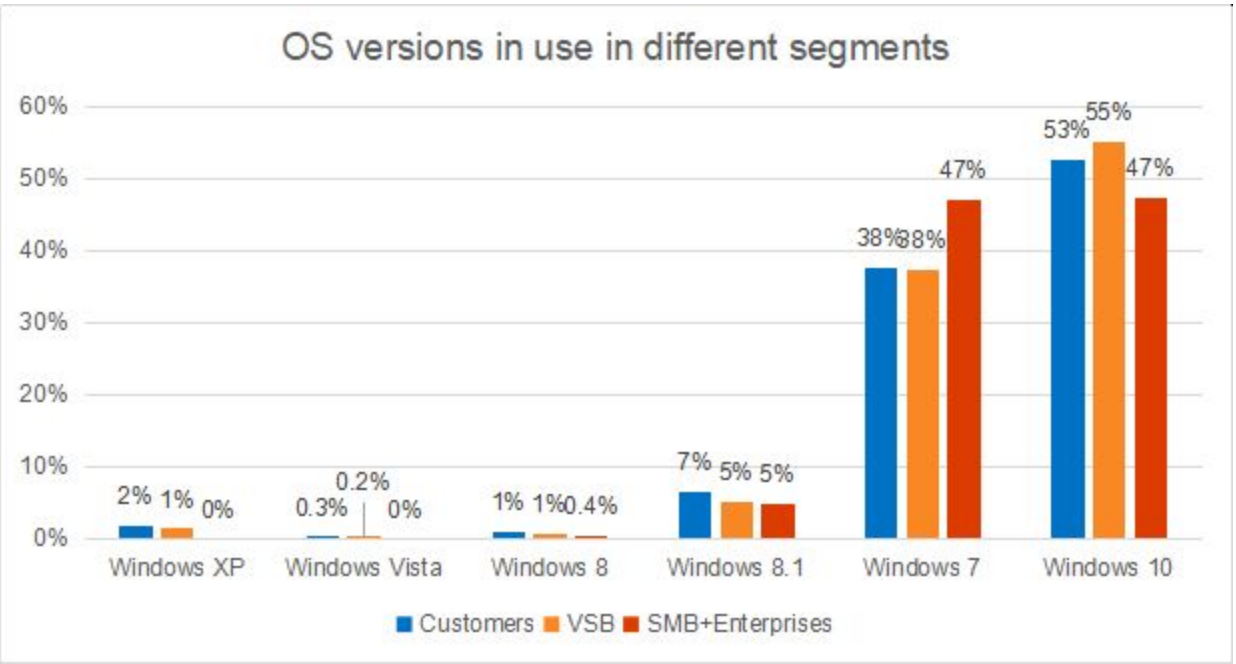
This is one of two TPM flaws which was just revealed by researchers this week, the other being an equivalent in Intel TPMs. Which, being the main topic of this week's podcast, we will be covering in full detail at the end of the episode.

But, speaking of Patch Tuesday...

We should note that only two Patch Tuesday's remain for Windows 7 and Server 2008 systems: this December 2019 and January 2020. After that… no more life support IV drip. This leaves end users with the choice to either bite the bullet and move to Windows 10, or go Commando and continue using Windows 7 and Server 2008 without any future updates.

I'll also note that once upon a time it would have made sense for the rational Windows 7 user to hold off for a while before adopting a new operating system. This would allow its "newness" to settle down and for most new problems to be found and fixed. But Windows 8 and 8.1 were widely viewed as a catastrophe, and it is now fully apparent that Microsoft has no plans to allow Windows 10 to ever "settle down". Windows 10 will continue to be an unsettled work-in-progress.

However, as we have observed, the vast majority of Windows 7 systems are not in the hands of end users, since most of them would have been unwittingly steam rolled by Microsoft's heavy handed push onto the Windows 10 platform. It's corporate users who have wisely and responsibly, in my opinion, chosen to remain on their trusty, stable, and working-just-fine-thank-you-very-much Windows 7 platforms. And it's these users who are being presented with choice #3: Start to pay for continued Windows 7 and Server 2008 updates through Microsoft's Extended Security Updates program.



https://techcommunity.microsoft.com/t5/Windows-IT-Pro-Blog/How-to-get-Extended-Security-Updates-for-eligible-Windows/ba-p/917807

CheckM8 & https://Checkra.in

Aside from the fact that, as we covered last week, the exploit for the Windows BlueKeep vulnerability has been perfected so that it no longer has a high incidence of BSOD's… arguably the most significant thing going on at the moment is the continuing work to perfect "CheckRa1n" the iOS CheckM8 BootROM vulnerability exploit. Since it's a big deal, I want to give it a bit of our time and check-in with where it stands today.

Last week's version history: 0.9 / 0.9.1 / 0.9.2
Since last week's podcast: 0.9.3 / 0.9.3.2 / 0.9.5

Lots of various bugs are being found and squashed.
The whole, inherently unauthorized, system is becoming increasingly stable.

Since I don't yet have any first-hand experience either with Jailbreaking or this particular emerging Jailbreaking system, I went looking for some interesting and pertinent feedback from those who have been playing with it so far. Dan Goodin, who covers security topics for ArsTechnica had also done some digging. I'll paraphrase from what he found and reported:

What are the benefits?

First, Checkra1n is extremely reliable and robust, particularly for a tool that's still in beta mode. It jailbreaks a variety of older iDevices quickly and reliably. It also installs an SSH server and other utilities, a bonus that makes the tool ideal for researchers and hobbyists who want to dig into the internals of their devices.

Ryan Stortz, an iOS security expert and principal security researcher at the firm Trail of Bits, said in an interview: "I expected it to be a little rougher around the edges for the first release. It's really nice to be able to install a new developer beta on your development iPhone and have all your tooling work out of the box. It makes testing Apple's updates much much easier."

Another benefit of Checkra1n is that it promises to work reliably on a wide array of hardware.

Also significant is that Checkm8-based jailbreaks will work forever on these devices. Unlike most jailbreaks, which exploit vulnerabilities in iOS, Checkm8 targets a flaw in the Boot ROM, which is the first code that runs when an iDevice is turned on. This code is burned into the hardware itself and cannot be patched. This is the reason Checkra1n will work with every new release of iOS over the lifetime of a vulnerable phone.

This means that people can continue to enjoy the benefits and security fixes available in new iOS releases without losing the ability to jailbreak their devices. As we know, until now, new versions of iOS inevitably fix any jailbreaking vulnerabilities that have been found and are being leveraged. This makes it a far cry from jailbreaks of the past which forced users to run an outdated version of iOS. The last time a jailbreak targeted the Boot ROM was in 2010 when hacker George Hotz (aka Geohot) developed one for the iPhone 3GS and iPhone 4.

Checkra1n is also helpful because it does not hide its presence. A large Checkra1n logo displays during bootup. And the home screen will include the Cydia and Checkra1n apps, neither of which appear when an iDevice runs normally.

And, as we know, as would be the case for any Checkm8-based jailbreaks, Checkra1n requires physical access to the vulnerable device and a reboot, which means user data and Touch ID and Face ID are inaccessible until the next time a PIN is entered to unlock the device. This renders remote exploits impossible.

So those were the benefits. What are the downsides?

(I should note beforehand that I don't consider any of these to be inherent limitations, since I've never endorsed the use of Jailbreaks for routine "I want my phone to have a different app launcher" style use.)

Dan writes: Checkm8-based jailbreaks, including Checkra1n, come with some notable limitations that many jailbreaking enthusiasts consider deal-breakers. First, Checkm8 doesn't work on iDevices introduced in the past two years, specifically those with A12 and A13 CPUs. That limits the jailbreak to older devices, most of which—but not all—are no longer sold in retail outlets. (Oh, boo hoo!)

The other major limitation is that Checkm8-based jailbreaks are "tethered," meaning they don't survive a reboot. Each time the device is restarted, it must first be connected to a Mac—eventually Windows versions of Checkra1n are expected—and jailbroken all over again. Untethered jailbreaks, by contrast, are much more popular because they allow iDevices to boot normally, without being connected to a computer each time.

Another drawback to any jailbreak is that it's an inevitably risky thing, since it unbinds an iDevice from the protections and quality assurances Apple has painstakingly built into iOS over more than a decade. Apple warns here that jailbreaking can "cause security vulnerabilities, instability, shortened battery life, and other issues." The stakes are raised further by the beta status of Checkra1n. The Checkra1n website warns: "This release is an early beta preview and as such should not be installed on a primary device. We strongly recommend proceeding with caution."

Then there are the risks of error by inexperienced users who are drawn to Checkra1n's reliability, robustness, and its promise to work—on older devices, anyway—in perpetuity.

Christoph Hebeisen, head of security research at mobile security provider Lookout, said: "The biggest threat from Checkra1n is how easily a non-technical user can jailbreak their device, which then leaves it vulnerable to additional attacks," One protection that Checkra1n deactivates is the iOS sandbox, which cordons off sensitive parts of iOS from the apps it runs. The risk is heightened by the ability of jailbroken devices to run any app. Normally iPhones and iPads can run only apps that are available in the App Store, which vets submissions for security and stability before allowing them in.

There's a more subtle threat posed by Checkra1n's ease in almost completely unpairing a device from the protections that have made iOS arguably the world's most secure OS. As noted earlier,

it would be hard for someone to use this jailbreak maliciously against someone else. But Ryan Stortz, the iOS security expert at Trail of Bits, notes that Checkra1n's existence creates a new threat where none existed before

Ryan noted: "The threat is more real now because a sophisticated exploit is available to everyone." He went on to theorize cases of attackers reverse-engineering Checkra1n and combining its jailbreaking capabilities with rootkits or other malicious code. In other words, Checkra1n opens the device up to other more nefarious exploits. And the only thing that attackers would need would be brief access to an iDevice. After that, attacker-installed malware could covertly steal text messages, login credentials, cryptographic keys, and any other sort of sensitive data. These attacks would be particularly effective against iPhones and iPads that don't use fingerprints or face scans for unlocking. He explained:

Checkm8 allows someone to undermine the trust of the iOS secure boot chain. Checkra1n makes it easy to do. It's true that checkra1n puts a nice logo on it and installs development tools, but that doesn't need to happen. Someone will modify checkra1n to remove the logo and install a rootkit instead. You'll pick up your phone after Checkra1n is surreptitiously installed and unlock it, allowing the rootkit full access to your personal data.

It has been possible to create this type of malicious jailbreak since late September, when the Checkm8 exploit became public. But that kind of attack required huge amounts of time and skill. That problem has now been solved and no one would bother to do that now that Checkra1n exists and appears to be so well built.

The take-away from the sort of scenario Ryan hypothesizes is this: for journalists, dissidents, and other high-value targets who use iOS devices and can afford to, it's best to use hardware that has an A12 or higher CPU. An iDevice introduced in the past two years will ensure that it's safe from Checkra1n-derived attacks at border crossings, in hotel rooms, or in other situations that involve brief separations.

For iOS users who can't afford a newer iPhone or iPad, using Touch ID or Face ID can lower the chances of malicious jailbreaks since users can be tipped off that something is amiss if the iDevice unexpectedly requires a PIN. And whenever the device has been out of the user's control, even briefly—or users suspect anything else is amiss—they should reboot it.

My take remains that this will likely have minimal direct impact upon the typical iDevice user. It opens up iOS and Apple's work to much more outside scrutiny than Apple has chosen to permit before now. Now, they have no say in the matter. It will be very likely that white hat hackers might discover previously unknown vulnerabilities affecting the user's of ALL of Apple's devices. So that will be all for the best. And equally probable is that black hats might discover new ways of subverting intact iOS devices. Only time will tell.

GitHub launches Security Lab to boost open source security
In happier news, last Thursday, during the GitHub Universe developer conference, GitHub announced "Security Lab" to create an open global platform for reporting and fixing security vulnerabilities in open source projects before they do serious damage.

In years past we've touched upon a few audits of important open source software. Early on, Matthew Green largely drive the effort to audit TrueCrypt. There have been a couple of audits of OpenSSL, and an audit of the new Unbound DNS server has recently been fully funded. But I think we would probably all agree that it's probably long past time for the creation of a public effort to incentivize and reward those who discover and report problems in open source software. We now have such a facility through GitHub.

- https://securitylab.github.com/
- https://github.blog/2019-11-14-announcing-github-security-lab-securing-the-worlds-code-together/

Jamie Cool explained the GitHub Security Lab mission as follows. He wrote:

We all share a collective responsibility to keep open source software secure—none of us can do it alone. Today at GitHub Universe, we announced GitHub Security Lab to bring together security researchers, maintainers, and companies across the industry who share our belief that the security of open source is important for everyone.

We're excited to have an initial set of partners that have all committed to achieving this goal. Together, we're contributing tools, resources, bounties, and thousands of hours of security research to help secure the open source ecosystem.

As part of today's announcement, GitHub Security Lab is making CodeQL freely available for anyone to find vulnerabilities in open source code. CodeQL is a tool many security research teams around the world use to perform semantic analysis of code, and we've used it ourselves to find over 100 reported CVEs in some of the most popular open source projects. (GitHub obtained CodeQL when they purchased its developer, Semmle.)

We're also launching the GitHub Advisory Database, a public database of advisories created on GitHub, plus additional data curated and mapped to packages tracked by the GitHub dependency graph.

GitHub's approach to security addresses the whole open source security lifecycle. GitHub Security Lab will help identify and report vulnerabilities in open source software, while maintainers and developers use GitHub to create fixes, coordinate disclosure, and update dependent projects to a fixed version.

GitHub Security Lab's mission is to inspire and enable the global security research community to secure the world's code. Our team will lead by example, dedicating full-time resources to finding and reporting vulnerabilities in critical open source projects. The team has already had over 100 CVEs issued for security vulnerabilities it has found.

Securing the world's open source software is a daunting task. First, there's scale: the JavaScript ecosystem alone has over one million open source packages. Then there's the shortage of security expertise: security professionals are outnumbered 500 to one by developers. Finally there's coordination: the world's security experts are spread across thousands of companies. GitHub Security Lab and CodeQL will help level the playing field.

Joining us in this effort are the following companies, donating their time and expertise to find and report vulnerabilities in open source software. Each have committed to contribute in a different way, and we hope others will join us in future.

> F5, Google, HackerOne, Intel, IOActive, J.P. Morgan, LinkedIn,
> Microsoft, Mozilla, NCC Group, Okta, Trail of Bits, Uber, VMWare

To empower the research community, we're also making our state-of-the art code analysis engine, CodeQL, free to use on open source. CodeQL lets you query code as though it were data. If you know of a coding mistake that caused a vulnerability, you can write a query to find all variants of that code, eradicating a whole class of vulnerabilities forever.

To get started with CodeQL: https://securitylab.github.com/tools/codeql

If you're a security researcher or work in a security team, we want your help. Securing the world's open source software will require the whole community to work together. GitHub Security Lab will run events and share best practices to help everyone participate. Follow the @GHSecurityLab account on Twitter for more details.

As the world's security researchers uncover more vulnerabilities, maintainers and end users need better tools to handle them. Today the process for addressing a new vulnerability is often ad hoc. Forty percent of new vulnerabilities in open source don't have a CVE identifier when they're announced, meaning they're not included in any public database. Seventy percent of critical vulnerabilities remain unpatched 30 days after developers have been notified.

We're fixing that. Maintainers and developers can now work together directly on GitHub to help ensure new vulnerabilities are only disclosed when maintainers are ready, and that developers can update to fixed versions quickly and easily.

This will be accomplished through GitHub Security Advisories. With Security Advisories, maintainers can work with security researchers on security fixes in a private space, apply for a CVE directly from GitHub, and specify structured details about the vulnerability. Then, when they're ready to publish the Security Advisory, GitHub will send security alerts to all affected projects.

Jaime notes that receiving a notification about vulnerable dependencies is helpful, but getting a pull request with a fix is even better. To help developers respond quickly to new vulnerabilities GitHub creates automated security updates—pull requests that update a vulnerable dependency to a fixed version.

Automated security updates were launched in beta at GitHub Satellite 2019 and are now generally available and rolled out to every active repository with security alerts enabled.

We've made all of the data that maintainers create in GitHub Security Advisories, plus additional data curated and mapped to packages tracked by the GitHub dependency graph, available for free. Explore the new GitHub Advisory Database in your browser, link directly to records with CVE identifiers in comments, or access the data programmatically using the Security Advisory API endpoint.

GitHub Advisory Database: https://github.com/advisories
Security Advisory API endpoint: https://developer.github.com/v4/object/securityadvisory/

Warrantless searches of devices at US borders were just ruled unconstitutional
We've discussed a few instances of this abuse in the past. Here are a couple of examples:

Take the case of natural-born US citizen Sidd Bikkannavar, a NASA engineer who was detained by US Customs and Border Protection in 2017 and pressured to hand over his NASA-issued phone and the PIN to get into it. This was for no cause and in spite of the fact that the work-issued phone could have contained sensitive information relating to his employment at the space agency, and in spite of the fact that NASA employees are obligated to protect all work-related information. The CBP officer returned his phone half an hour later saying that it had been searched using "algorithms".

Or how about the artist Aaron Gach, another natural-born US citizen who was forced to unlock his phone after returning from putting on a gallery installation in Brussels. That installation focused on "mass incarceration, government control, and political dissent".

Or Diane Maye, a college professor and retired US Air Force officer who was detained for two hours at Miami International Airport upon returning home from a vacation in Europe. At the time that the lawsuit was filed in 2017, Maye said that the encounter left her feeling "humiliated and violated." She explained that she worried that border officers -- again, without cause or reason -- would read her email messages and texts, and look through her photos. She said that this was her life, and a border officer held it in the palm of his hand. She joined this lawsuit because she strongly believe the government shouldn't have the unfettered power to invade our privacy without probable cause.

For the past 10 years, since 2009, U.S. border agents have been legally allowed to search electronic devices of travelers at borders without any specific cause or suspicion in the interest of protecting borders and enforcing border security.

Then in 2016, that law was revised to require "reasonable suspicion" for anything beyond basic searches. But that still allowed agents to require travelers to unlock phones and gave them free rein to read messages and other basic information.

So, last Tuesday, a federal court in Boston ruled that suspicion-free, warrantless searches of travelers' electronic devices at US border entry points are unconstitutional. The decision comes out of a lawsuit – Alasaad v. McAleenan – filed against the Department of Homeland Security (DHS) in 2017 by the American Civil Liberties Union (ACLU) and the EFF on behalf of 11 travelers: 10 legal residents of the US and one lawful permanent resident, all of whom were forced into warrantless searches of their mobile phones and laptops at the border (including the three I mentioned).

Sophia Cope, a senior staff attorney with the Electronic Frontier Foundation (EFF) said of this ruling: "This is a great day for travelers who now can cross the international border without fear that the government will, in the absence of any suspicion, ransack the extraordinarily sensitive information we all carry in our electronic devices."

However, it remains legal for border agents to look through the devices of travelers who get referred for a secondary inspection. During the primary inspection, travel documents and passports are reviewed. If a secondary inspection is needed, officers may then search phones, thumb drives, computers and other electronic devices to determine whether they should let somebody into the country or to identify potential legal violations.

According to the ACLU, the Boston district court's order puts an end to the authority that CBP and ICE had been granting themselves to search and seize travelers' devices for purposes beyond enforcing immigration and customs laws. At this point, border officers are required to demonstrate "individualized suspicion of contraband" before they can search a traveler's device.

Today, unlike before the revolution of the smartphone, we are all carrying what amounts to a database snapshot of our lives in our pockets. The fact that it CAN be provided doesn't mean that it SHOULD or MUST be provided upon demand. It is our personal data and information. So this feels to me like the right outcome. I'm sure that the border agents are truly doing what they think they're supposed to. They likely worry "what if I let this suspicious-looking person through without inspecting their devices as I'm told I have the right to do... and then they do something horrific and I'm asked why I didn't give them a full "electronic frisking"?? So, if law is clarified so that NO border agents are permitted the discretion to electronically frisk their entrants, that problem of "whose responsibility is it" is sidestepped.

Another WhatsApp bug lets hackers quietly install spyware on your device

Just so we don't consider these problems just another in a long line of "theoretical" vulnerabilities, let's remember that Israel's NSO Group was recently found to have leveraged another recent WhatsApp vulnerability, in its VoIP calling, to successfully install Pegasus spyware on nearly 1400 targeted Android and iOS devices worldwide. So... remote code execution flaw in WhatsApp won't go unexploited for long.

And in October we noted that WhatsApp was using an open-source GIF image parsing library which contained a double-free memory corruption bug which could be leveraged for remote code execution in the context of the WhatApps application. It was discovered by Vietnamese security researcher Pham Hong Nhat in May.

So now we have the news that last month WhatApp quietly patched another critical vulnerability that would have allowed attackers to remotely compromise targeted devices to steal secured chat messages and files stored on them. The vulnerability — which was tracked as CVE-2019-11931 — is a stack-based buffer overflow issue that resided in the way WhatsApp was parsing the elementary stream metadata of an MP4 file, resulting in either denial-of-service or remote code execution attacks.

As we've often seen, parsing stream metadata, especially for a compressed file format, is very very difficult to make absolutely robust. When decoding the meetadata it's SO EASY to make the assumption that the sender was a valid codec.

In any event, to remotely exploit the vulnerability, all an attacker needs is the phone number of a targeted user and to then send them a maliciously crafted MP4 file over WhatsApp. It can then silently install a malicious backdoor or spyware app on the now-compromised device. The vulnerability affects both consumers as well as enterprise apps of WhatsApp for all major platforms, including Google Android, Apple iOS, and Microsoft Windows.

According to an advisory published by Facebook the list of affected app versions are as follows:

- Android versions before 2.19.274
- iOS versions before 2.19.100
- Enterprise Client versions before 2.25.3
- Windows Phone versions before and including 2.18.368
- Business for Android versions before 2.19.104
- Business for iOS versions before 2.19.100

So, you'll want to be sure that you're using the latest updated version of WhatsApp. It's not clear whether the MP4 vulnerability was exploited as a 0-day in the wild before Facebook learned about and patched it.

But if you consider yourself to be a potential surveillance target and you may have received a random and unexpected MP4 video file over WhatsApp from an unknown number in recent months, you might want to consider whether something evil might have crawled into your device.

ZombieLoad v2
https://zombieloadattack.com/

Last May we had fun discussing ZombieLoad, one of a group of several newly discovered additional attacks against Intel processors. In that sense, ZombieLoad and its new cousins were similar to the original Meltdown, Spectre & Foreshadow exploits, but because they leverage a different region of the Intel architecture, these new ones were referred to as Microarchitectural Data Sampling, or MDS, flaws and exploits. At the time, ZombieLoad was deemed the most dangerous of the new MDS attacks (the others being Fallout and RIDL) because it could be used to retrieve more information than the others.

However, there was always a 5th MDS attack which researchers agreed to keep secret because Intel had no solution for it. Now, with Intel's release of new firmware microcode we're learning of it for the first time. It's being called Zombiload v2 (CVE-2019-11135), because it's a variation of the original Zombieload v1 vulnerability, but this one works against Intel's newer CPUs, which, sadly, Intel had claimed had protections against speculative execution attacks baked in at the hardware level. But... not so much.

An updated release of the Zombieload research paper reveals that the Zombieload v2 attack exploits the Intel Transactional Synchronization Extensions (TSX) Asynchronous Abort operation which is invoked when an attacker creates a conflict between read operations inside a CPU. The read conflict for TSX Asynchronous Abort (TAA) leaks data about what's being processed inside the CPU.

In the revised version of their whitepaper the research team wrote: "The main advantage of this approach is that it also works on machines with hardware fixes for Meltdown, which we verified on an i9-9900K and Xeon Gold 5218."

The only condition for a Zombieload v2 attack is that the targeted CPU supports the Intel TSX instruction-set extension, which the research team said is available by default in all Intel CPUs sold since 2013. The first Intel CPU series to feature TSX support was Haswell. And everything that came after is affected. Intel's Cascade Lake, which the company released in April this year, was supposed to be the company's first product that featured protections against side-channel and speculative execution attacks at the hardware level... but it, too, is affected by ZombieLoad v2.

Intel has released firmware and both the Windows and Linux kernels are responding.
https://support.microsoft.com/en-us/help/4530989/guidance-for-protecting-against-intel-processor-machine-check-error

Microsoft: "On November 12, 2019, Intel published a technical advisory around Intel® Processor Machine Check Error vulnerability that is assigned CVE-2018-12207. Microsoft has released updates to help mitigate this vulnerability for guest Virtual Machines (VMs) but the protection is disabled by default. Enabling this protection requires an action on the Hyper-V hosts running untrusted VMs. Follow the guidance in the "Registry setting" section to enable this protection on the Hyper-V hosts running untrusted VMs."

Given that not a single instance of any one of these low-yield theoretical attacks has EVER been made usefully practical nor ever encountered in the wild, and given that the only true vulnerability is virtual shared hosting environments where a targeted attack might be attempting to break cross OS isolation, this really should not be a concern for any end user.

The ByteCode Alliance
https://bytecodealliance.org/

Mozilla, Fastly, Intel, RedHat

WebAssembly is outgrowing its browser.

As we know, there was some early competition among competing JavaScript replacements to increase client-side web application performance. WebAssembly won. And it is now supported by all major web browsers and has proven to be the right choice.

The ByteCode Alliance is the next effort to generalize WebAssembly into a generic platform and architecture agnostic common runtime suitable for use on everything from super-lean IoT widgets through high-end CDN server farms. In other words... completely outside the browser. If this sounds familiar it's the role that Sun had always hoped JAVA might obtain.

However, what's different from JAVA is that WebAssembly has its Internet-facing origin in the browser, where strong attack-hardening was explicit. So security is built-in from the start. In his introduction over on Mozilla, Lin Clark wrote:

"We have a vision of a WebAssembly ecosystem that is secure by default, fixing cracks in today's software foundations. And based on advances rapidly emerging in the WebAssembly community, we believe we can make this vision real.

We're already putting these solutions to work on real world problems, and those are moving towards production. But as an Alliance, we're aiming for something even bigger…

As an industry, we're putting our users at risk more and more every day. We're building massively modular applications, where 80% of the code base comes from package registries like npm, PyPI, and crates.io.

Making use of these flourishing ecosystems isn't bad. In fact, it's good!

The problem is that current software architectures weren't built to make this safe, and bad guys are taking advantage of that… at a dramatically increasing rate.

What the bad guys are exploiting here is that we've gotten our users to trust us. When the user starts up your application, it's like the user's giving your code the keys to their house. They're saying "I trust you".

But then you invite all of your dependencies, giving each one of them a full set of keys to the house. These dependencies are written by people who you don't know, and have no reason to trust.

As a community, we have a choice. The WebAssembly ecosystem could provide a solution here… at least, if we choose to design it in a way that's secure by default. But if we don't, WebAssembly could make the problem even worse.

As the WebAssembly ecosystem grows, we need to solve this problem. And it's a problem that's too big to solve alone. That's where the Bytecode Alliance comes in.

The Bytecode Alliance is a group of companies and individuals, coming together to form an industry partnership.

Together, we're putting in solid, secure foundations that can make it safe to use untrusted code, no matter where you're running it—whether on the cloud, natively on someone's desktop, or even on a tiny IoT device.

With this, developers can be as productive as they are today, using open source in the same way, but without putting their users at risk.

This common, reusable set of foundations can be used on their own, or embedded in other libraries and applications.

So, this is going to be a very interesting effort to keep our eyes on. Our listeners have heard me bemoan many times in the past that the way code is still be created today is the wild west, and

that we need to rethink the way we create apps at a fundamental level. If nothing else it's clear that these guys get that. They are saying all the right things. We'll see where this leads.

Miscellany

- Disney+ "The Mandalarian" ... Remember that “Tricks is for kids”... so is Mandalarian.
- "Ford vs Ferrari" -- the rare perfect movie.

TPM-FAIL

http://tpm.fail/

Given that we now have the DOT FAIL TLD, I have the feeling we're going to be seeing many more *.FAIL vulnerability and exploit names.

http://tpm.fail/tpmfail.pdf

ABSTRACT: Trusted Platform Module (TPM) serves as a hardware-based root of trust that protects cryptographic keys from priv-ileged system and physical adversaries. In this work, we per-form a black-box timing analysis of TPM 2.0 devices deployed on commodity computers. Our analysis reveals that some of these devices feature secret-dependent execution times during signature generation based on elliptic curves. In particular, we discovered timing leakage on an Intel firmware-based TPM as well as a hardware TPM. We show how this information allows an attacker to apply lattice techniques to recover 256-bit private keys for ECDSA and ECSchnorr sig-natures. On Intel fTPM, our key recovery succeeds after about 1,300 observations and in less than two minutes. Similarly, we extract the private ECDSA key from a hardware TPM manu-factured by STMicroelectronics, which is certified at CommonCriteria (CC) EAL 4+, after fewer than 40,000 observations. We further highlight the impact of these vulnerabilities by demonstrating a remote attack against a StrongSwan IPsecVPN that uses a TPM to generate the digital signatures for authentication. In this attack, the remote client recovers the server’s private authentication key by timing only 45,000authentication handshakes via a network connection.

The vulnerabilities we have uncovered emphasize the dif-ficulty of correctly implementing known constant-time tech-niques, and show the importance of evolutionary testing and transparent evaluation of cryptographic implementations.Even certified devices that claim resistance against attacks require additional scrutiny by the community and industry, as we learn more about these attacks.

Okay… so in the first place, unlike all of the previous Intel CPU leakage attacks, these are PRACTICAL to accomplish and, as they demonstrated, they can be successfully performed remotely over a relatively high-speed (thus low latency and low-jitter) network.

Intel was initially not convinced of the attack's severity, so the researchers demonstrated a successful more attack over a network and that really got Intel's attention.

Intel has a firmware implementation of the TPM within what Intel calls their "Platform Trust Technology." The bad news is that it's the easiest of the attacks to exploit. The good news is that, being implemented in firmware, it can be patched and updated. So the big takeaway here is that anyone who is truly relying upon their TPM technology to protect their secrets will need to obtain an update motherboard BIOS which incorporates these fixes. And it might also be that we'll see an update from Microsoft and for Linux OSes that patches the Intel firmware on-the-fly.

The research team intends to publish the tools they used to analyze the vulnerable TPMs, along with proof-of-concept code, on GitHub. This mixed blessing will allow system administrators to determine which TPMs they are using on their many various systems. So the proof-of-concept code should help these sysadmins test and see if they have devices vulnerable to the two attacks.

The downside is that the same proof-of-concept code will likely end up helping attackers, once it gets published online. Therefore, obtaining and applying the Intel Protected Trust Technology firmware updates should be a top priority for enterprises with this exposure.

On the hardware side, there are many manufacturers of hardware Trusted Platform Modules. The researchers tested all that they could find and, for example, those by Infineon and Nuvoton were found not to have secret-based timing influences... but the highly popular TPM by STMicrosystems WAS found to alter its response timing as a function of the secrets it was manipulating. In other words, it's not operating in "constant time". The good news is, the timing variations are so slight that many many more tests are required, and that also renders this attack impractical over any network since network packet jitter would completely hide the timing leakage.

But... it CAN still be exploited by any local attacker who can run code on the platform. So, if important secrets are stored in the system's TPM -- such as anyone who is depending upon Bitlocker for their full disk encryption -- then this might be of some concern. The bad new is that the flaw is in hardware. STMicro has updated their hardware for the future, but someone who has a vulnerable STMicro TPM soldered into their motherboard is out of luck.