Security Now! #759 - 03-24-20

TRRespass

This week on Security Now!

This week we look at a new unpatched 0-day attack affecting billions of Windows users, Mozilla's reversal on TLS 1.0 and 1.1 deprecation due to the coronavirus, a welcome micropatch for Win7 and Server 2008, Chrome’s altered release schedule during coronavirus, AVAST's latest screw-up, a new threat affecting Android users, the results from last week's Pwn2Own competition, and few observations about the coronavirus math and some worthwhile explainer videos. Then we look at where we are with RowHammer after 6 years.

Security News

Two new un-patched 0-days affecting billions of Windows users

Microsoft's security advisory was published yesterday, March 23rd, titled: "ADV200006 | Type 1 Font Parsing Remote Code Execution Vulnerability"

https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV200006

> Microsoft is aware of limited targeted attacks that could leverage un-patched vulnerabilities in the Adobe Type Manager Library, and is providing the following guidance to help reduce customer risk until the security update is released.
>
> Two remote code execution vulnerabilities exist in Microsoft Windows when the Windows Adobe Type Manager Library improperly handles a specially-crafted multi-master font - Adobe Type 1 PostScript format.
>
> There are multiple ways an attacker could exploit the vulnerability, such as convincing a user to open a specially crafted document or viewing it in the Windows Preview pane.
>
> Microsoft is aware of this vulnerability and is working on a fix. Updates that address security vulnerabilities in Microsoft software are typically released on Update Tuesday, the second Tuesday of each month. This predictable schedule allows for partner quality assurance and IT planning, which helps maintain the Windows ecosystem as a reliable, secure choice for our customers. The operating system versions that are affected by this vulnerability are listed below. Please see the mitigation and workarounds for guidance on how to reduce the risk.

Both of these unpatched flaws are being used in limited, targeted attacks and impact all supported versions of the Windows operating system—including Windows 10, 8.1 and Server 2008, 2012, 2016, and 2019 editions, as well as Windows 7 for which, as we know, Microsoft ended support after this past January.

The vulnerabilities reside in the Windows Adobe Font Type Manager Library, which is a font parsing and display subsystem used by Windows Explorer to display the content of a file in the 'Preview' or 'Details' panes without users needing to open it. It is also used by many pieces of 3rd-party software. The flaws results from Adobe Type Manager Library "improperly handling a specially-crafted multi-master font - Adobe Type 1 PostScript format," which then allowing remote attackers to execute arbitrary malicious code on targeted systems. The obvious attack route involves convincing a user to open a specially crafted document or viewing it in the Windows Preview pane. It's not clear whether the flaws can also be triggered remotely over a web browser by convincing a user to visit a web-page containing specially-crafted malicious OTF fonts, and there are multiple other ways an attacker might be able to exploit the vulnerability, such as through the Web Distributed Authoring and Versioning (WebDAV) client service.

The Microsoft advisory wording strongly suggests that despite the fact that essentially every Windows user is vulnerable and will be vulnerable until April 14th (another latest-possible patch Tuesday) when this is patched, they do not intend to do anything about it for the next three weeks. So... We're on our own.

In the meantime, Microsoft IS offering various workarounds:

They wrote: Disabling the Preview and Details panes in Windows Explorer prevents the automatic display of OTF fonts in Windows Explorer. While this prevents malicious files from being viewed in Windows Explorer, it does not prevent a local, authenticated user from running a specially crafted program to exploit this vulnerability. To disable these panes in Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, and Windows 8.1, perform the following steps:

- Open Windows Explorer, click Organize, and then click Layout.
- Clear both the Details pane and Preview pane menu options.
- Click Organize, and then click Folder and search options.
- Click the View tab.
- Under Advanced settings, check the Always show icons, never thumbnails box.
- Close all open instances of Windows Explorer for the change to take effect.

After April 16th, when this has been fixed, simply turn off the “Always show icons” checkbox.

The fact that this doesn't prevent a local user from exploiting the vulnerability suggests that this might be an ongoing way to hack into any Windows 7 and Server 2008 machine using an unprivileged account. This will presumably never be fix for the large population of Win7 and Server 2008 machines that continue running. So this serves as another example of the growing list of vulnerabilities that aging systems will have that will never be repaired. Somewhere those collecting cyber assault tools are making a note of this for the time they need to hack into a Win7 or Server 2008 machine.

Microsoft also recommends disabling the Windows WebClient service. They wrote:

Disabling the WebClient service helps protect affected systems from attempts to exploit this vulnerability by blocking the most likely remote attack vector through the Web Distributed Authoring and Versioning (WebDAV) client service. After applying this workaround it is still possible for remote attackers who successfully exploit this vulnerability to cause the system to run programs located on the targeted user's computer or the Local Area Network (LAN), but users will be prompted for confirmation before opening arbitrary programs from the Internet.

To disable the WebClient Service, perform the following steps:

- Click Start, click Run (or press the Windows Key and R on the keyboard), type Services.msc and then click OK.
- Right-click WebClient service and select Properties.
- Change the Startup type to Disabled. If the service is running, click Stop.
- Click OK and exit the management application.

The final solution is the only one I would consider useful, which is to rename the ATMFD.DLL. That way it simply ceases to exist for whatever other code seeks to load and use its fictions. Microsoft details the process for both 32 and 64 bit Windows 10:

32-bit:

```
cd "%windir%\system32"
takeown.exe /f atmfd.dll
icacls.exe atmfd.dll /save atmfd.dll.acl
icacls.exe atmfd.dll /grant Administrators:(F)
rename atmfd.dll x-atmfd.dll
```

64-bit:

```
cd "%windir%\system32"
takeown.exe /f atmfd.dll
icacls.exe atmfd.dll /save atmfd.dll.acl
icacls.exe atmfd.dll /grant Administrators:(F)
rename atmfd.dll x-atmfd.dll

cd "%windir%\syswow64"
takeown.exe /f atmfd.dll
icacls.exe atmfd.dll /save atmfd.dll.acl
icacls.exe atmfd.dll /grant Administrators:(F)
rename atmfd.dll x-atmfd.dll
```

Then restart the system.

With Windows 8.1 and earlier this can be done with the registry:

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows]
"DisableATMFD"=dword:00000001
```

The end of the advisory offered a Q&A section. Most of the questions were about whether ESU would be needed any why wouldn't Windows 7 be fixed. But the final three questions were worth sharing with our listeners:

Q: Is the Outlook Preview Pane an attack vector for this vulnerability?
A: No, the Outlook Preview Pane is NOT an attack vector for this vulnerability
Q: Is the Windows Explorer Preview Pane an attack vector for this vulnerability?
A: Yes, the Windows Preview Pane is an attack vector for this vulnerability
Q: Is Enhanced Security Configuration, which is on by default on Windows Servers, a mitigation for this vulnerability?
A: No, Enhanced Security Configuration does not mitigate this vulnerability.

Mozilla reversed itself on TLS v1.0 and 1.1 deprecation... due to the coronavirus.
As we know, TLS 1.0 and TLS 1.1 support was going to be dropped on March 10th, with the release of Firefox 74.0, to improve the security of website connections so that sites not supporting TLS 1.2 or 1.3 would show a "Secure connection failed" error page.

But then Coronavirus happened and the balance tipped just enough...

https://www.mozilla.org/en-US/firefox/74.0/releasenotes/#note-788289

> ~~We have disabled TLS 1.0 and TLS 1.1 to improve your website connections. Sites that don't support TLS version 1.2 will now show an error page.~~
>
> **We reverted the change for an undetermined amount of time to better enable access to critical government sites sharing COVID19 information.**

As we know, TLS 1.0 and TLS 1.1 support was going to be dropped on March 10th, with the release of Firefox 74.0, to improve the security of website connections so that sites not supporting TLS 1.2 or 1.3 would show a "Secure connection failed" error page.

But then Coronavirus happened and the balance tipped just enough...
Remember that the UK's Netcraft survey site reported that at the beginning of March 2020, over 850,000 websites are still using TLS v1.0 or 1.1. There are apparently some government sites among those, and Mozilla noticed that their Firefox users were being denied access.

In other times, this might have served to place some nice update pressure upon those sites. But given that getting information out to their visitors is more important, the pressure can wait.

A micropatch for Win7 and Server 2008
On patch Tuesday of this month we learned of CVE-2020-0881, which is an RCE (remote code execution) vulnerability affecting all versions of Windows.

https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-0881

A remote code execution vulnerability exists in the way that the Windows Graphics Device Interface (GDI) handles objects in the memory. An attacker who successfully exploited this vulnerability could take control of the affected system. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.

There are multiple ways an attacker could exploit the vulnerability:
☐ In a web-based attack scenario, an attacker could host a specially crafted website that is designed to exploit the vulnerability and then convince users to view the website. An attacker would have no way to force users to view the attacker-controlled content. Instead, an attacker would have to convince users to take action, typically by getting them to open an email attachment or click a link in an email or instant message.
☐ In a file-sharing attack scenario, an attacker could provide a specially crafted document file that is designed to exploit the vulnerability, and then convince users to open the document file.

The security update addresses the vulnerability by correcting the way that the Windows GDI handles objects in the memory.

So here we are again with a potentially exploitable flaw in Win7 and Server 2008 which Microsoft will not be patching. If nothing is done about this the flaw, it will persist and it will get added to the cyber attack inventory of state-actor and freelance hackers for use whenever they wish to leverage old and known -- but never patched -- Windows 7 and Server 2008 vulnerabilities.

The 0Patch guys have generated another of their micropatches to fix this problem for those who cannot obtain the official fix from Microsoft. It's available to 0Patch's paying customers ($26/year/workstation) and in this case it repairs the memory corruption issue in Windows GDI+ by adding a similar code block to the one Microsoft used in their official security fix. If this fix is missing, a carefully crafted exploit could lead to the deletion of a chosen data structure and then to a subsequent use-after-free exploit. On systems where the micro-patch is present, it implements a logically identical check, but also records an exploitation attempt event before redirecting execution flow to the safe path. The patch for 32-bit systems contains 4 instructions and the patch for 64-bit systems contains 5.

Here's a video of the exploit first without the 0patch enabled and then with it enabled:
https://youtu.be/kO38ienCDzk

https://0patch.com/pricing.html

For those who wish to remain on Windows 7 -- and, boy, that's looking smarter and smarter all the time -- it might be worth taking them up on their free trial and see what you think.

Chrome's release schedule has been impacted by the coronavirus

Last Wednesday the 18th, Google announced that major Chrome Browser and OS releases will be placed on hold due to the adjusted work schedules of employees having to work from home during the coronavirus sequestration. Their announcement stated that they would be placing priority on security, which makes sense. As we know, most new versions bring us a mixture of new features (or often these days the removal of some creaky old features) and, of course, security-related bug fixes.

So, during this work-at-home time, the Google Chrome development team will be working remotely and prioritize security updates that will be released as Chrome v80 updates. Google Chrome 80.0.3987.149 was released right after the company announced that Chrome v81 was delayed, with security fixes which patched 13 high severity vulnerabilities.

So... v81 is now on the back burner. It was originally slated to start rolling out last Tuesday on March 17th according to a post initially published on the Google Developers blog. That new Chrome release should have included support for form elements featuring a modernized appearance, hit testing for augmented reality, app icon badge support, and initial support for Web NFC.

Also affected were Android developers who were informed that they could now expect to experience longer than normal app review times due to the adjusted work schedules -- as many as 7 days or longer. A Google spokesman wrote: "Due to adjusted work schedules at this time, we are currently experiencing longer than usual review times. While the situation is currently evolving, app review times may fluctuate, and may take 7 days or longer."

Avast emergency-disables their internal JavaScript emulator

We haven't heard much from Google's Tavis Ormandy for a while. But earlier this month he used a nifty tool he developed back in 2017. We've talked about it before. It allows Tavis to run Windows DLLs under Linux where automated fuzzing and other security tests can be performed sort of "in vitro" as opposed to "in vivo." It's a very cool concept.

In this instance, this allowed Tavis to discover a trivially executed critical flaw in AVAST's A/V system. Tavis informed AVAST, whereupon after a week of scrambling around, they finally disabled this important component of their A/V product.

Tavis discovered a security flaw in Avast's JavaScript emulator engine which analyzes incoming JavaScript code for malicious action before allowing it to execute in browsers or email clients.

Tavis wrote:

> *Despite being highly privileged and processing untrusted input by design, it is unsandboxed and has poor mitigation coverage. Any vulnerabilities in this process are critical, and easily accessible to remote attackers.*

Tavis explained that exploitation of the bug was trivial once it was known. All it takes is sending a malicious JS or WSH file to a user via email, or tricking a user to access a boobytrapped file containing malicious JavaScript code. The faulty AVAST engine would then stumble over the file and the attacker would obtain SYSTEM-level access without any restrictions. Attackers would have the ability to then install malware on an Avast user's device.

A week passed after Tavis notified AVAST during which they did not appear to be taking any action. They were likely hoping for a quick fix. But the problems might be systemic. Whatever the case, after a week they decided to completely disable their product's JavaScript scanner until it could be fixed.

They issued the following statement:

> *Last Wednesday, March 4, Google vulnerability researcher Tavis Ormandy reported a vulnerability to us affecting one of our emulators. The vulnerability could have potentially been abused to carry out remote code execution.*
>
> *On March 9, he released a tool to greatly simplify vulnerability analysis in the emulator.*
>
> *We have fixed this by disabling the emulator, to ensure our hundreds of millions of users are protected from any attacks. This won't affect the functionality of our AV product, which is based on multiple security layers.*

There is no current timeline for when a patch would be ready.

So, here we are again, Leo, seeing another instance where a 3rd-party A/V add-on adds exploitable vulnerabilities where none existed before. And we've seen that 3rd-party A/V is often behind failures of the Windows Update system. It's just too invasive.

CookieThief - "FireSheep evolves for the 21st century"
https://securelist.com/cookiethief/96332/

Kaspersky's security announcement was titled: "CookieThief: a cookie-stealing Trojan for Android"

A combination of new modifications to Android malware code has given rise to Trojans able to steal browser and app cookies from compromised devices. On Thursday March 12th, 12 days ago, researchers from Kaspersky revealed a new malware family which, thanks to its actions, they dubbed "CookieThief." CookieThief uses a combination of exploits to acquire root rights to an Android device which it then uses to steal the user's Facebook cookie data.

Kaspersky wrote:

> We recently discovered a new strain of Android malware. The Trojan (detected as: Trojan-Spy.AndroidOS.Cookiethief) turned out to be quite simple. Its main task was to acquire root rights on the victim device, and transfer cookies used by the browser and the Facebook app to the cybercriminals' server. This abuse technique is possible not because of a vulnerability in the Facebook app or the browser themselves. Malware could steal cookie files of any website from other apps in the same way and achieve similar results.

As we know, cookie stealing is all about account hijacking. The only way in which a user's logged-on session is maintained is with one or more secret cookies which, these days, are marked 'secure' so that it will only be sent in TLS browser queries. No more simple-minded FireSheep which simply snagged the cookies over the HTTP channel after the user's connection reverted to HTTP following logon.

But simple static cookies represent a big vulnerability. Unlike any more dynamic system (like, well, SQRL, which always requires the client to solve a cryptographic problem and to return its solution with every query, thus reproving its identity), cookies are just static blobs of data. Although they might be changed periodically, they are simply regurgitated by the browser, which always returns the most recently set cookies for the domain.

The emergence of OAuth, which uses one's logged-on status at one site to authenticate the user's identity at another site means that, someone obtaining a user's "Facebook" cookies also obtains the ability to impersonate them -- since they appear to be validly logged onto Facebook. This would allow them to logon-on as that user at other sites which offer a "Logon with Facebook" option, thus spread the impersonation much farther.

Kaspersky wrote that:

> On the C&C server we also found a page advertising services for distributing spam on social networks and messengers, so it was not difficult to guess the motive behind the cookie-theft operation.
>
> But there's still a hurdle for the spammers that prevents them from gaining instant access to accounts just like that. For example, if Facebook detects an atypical user activity, the account may be blocked.
>
> However, during our analysis of Cookiethief, we uncovered another malicious app with a very

similar coding style and the same C&C server. The second "product" from (presumably) the same developers (detected as: Trojan-Proxy.AndroidOS.Youzicheng) runs a proxy on the victim's device.

We believe that Youzicheng is tasked with bypassing the security systems of the relevant messenger or social network using a proxy server on the victim's device. As a result, cybercriminals' request to the website will look like a request from a legitimate account and not arouse suspicion.

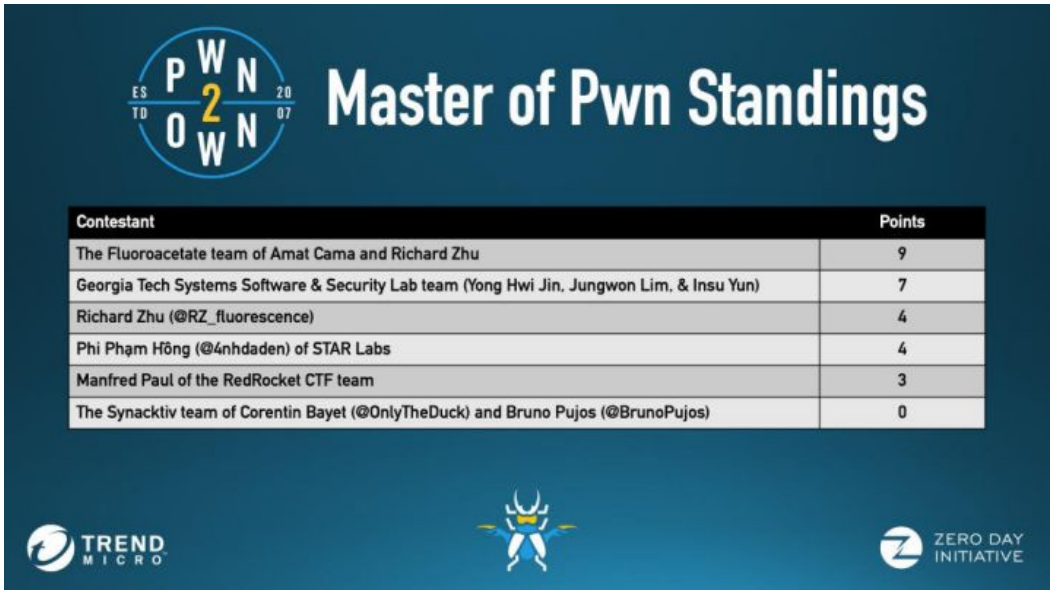
To implement this method, an executable file is first downloaded.
Then the proxy configuration is requested.
The downloaded file is then run.

By combining these two attacks, cybercriminals can gain complete control over the victim's account and not raise any suspicion from Facebook. These threats are only just starting to spread, and the number of victims, according to our data, does not exceed 1000, but the figure is growing.

PwnToOwn Spring 2020

Last Friday was the second and final day of the 2020 spring edition of HackerOne's always interesting Pwn2Own hacking contest.

This year's winner is (perhaps unsurprisingly) Team Fluoroacetate formed by security researchers Amat Cama and Richard Zhu. They won the contest after accumulating nine points across the two-day competition and extending their dominance by winning their fourth tournament in a row.



Contestant	Points
The Fluoroacetate team of Amat Cama and Richard Zhu	9
Georgia Tech Systems Software & Security Lab team (Yong Hwi Jin, Jungwon Lim, & Insu Yun)	7
Richard Zhu (@RZ_fluorescence)	4
Phi Phạm Hông (@4nhdaden) of STAR Labs	4
Manfred Paul of the RedRocket CTF team	3
The Synacktiv team of Corentin Bayet (@OnlyTheDuck) and Bruno Pujos (@BrunoPujos)	0

But, we're not currently meeting face-to-face and the spring edition of the Pwn2Own hacking contest has always been held during the CanSecWest cyber-security conference, occurring each spring in Vancouver, Canada. But the ongoing coronavirus outbreak has messed up many security researcher's travel who either could not or would not travel to Vancouver and potentially

put their health at risk. So, this year's Pwn2Own has become the first-ever hacking contest hosted virtually.

Although this sacrifices some of the in-person drama, the participants sent their exploits to Pwn2Own organizers in advance, who then ran the code during a live stream with all participants present... so much of the breath holding was preserved.

During the competition's two-day schedule, six teams managed to hack apps and operating systems like Windows, macOS, Ubuntu, Safari, Adobe Reader, and Oracle VirtualBox. All bugs exploited during the contest were immediately reported to their respective companies.

Okay, so what happened?

The first exploit of the first day was attempted by the Georgia Tech Systems Software & Security Lab team who targeted Apple Safari with a macOS kernel escalation of privilege. (So, browser-based kernel escalation attack. Yikes!)

The exploit succeeded when the Georgia Tech team used a six-bug exploit chain (that just makes me shake my head) to pop-up the calculator app on macOS and escalate its access rights to root. They earned a well-deserved $70,000 USD and 7 Master of Pwn points.

The second exploit attempt was launched by Flourescence (Richard Zhu by himself) who targeted Microsoft Windows with a local privilege escalation... and succeeded: He used a use-after-free vulnerability in Windows to escalate his local privilege and earned $40,000 USD and 4 points towards Master of Pwn.

The third exploit was brought by Manfred Paul of the RedRocket CTF team who targeted Ubuntu Desktop with a local privilege escalation. It was also successful. This newcomer to Pwn2Own used an improper input validation bug to escalate privileges and earned $30,000 and 3 Master of Pwn points.

For the 4th exploit, the Fluoroacetate team set their sights on Microsoft Windows with a local privilege escalation. They succeeded by leveraging a use-after-free bug in Windows to escalate themselves to full SYSTEM privilege. And they earned $40,000 and 4 Master of Pwn points.

The second day began when STAR Labs targeted Oracle’s VirtualBox for the first exploit in the Virtualization category... and it worked. The researcher used an out-of-bounds read bug for an info leak and an uninitialized variable for code execution on the VirtualBox hypervisor. He took home $40,000 and 4 Master of Pwn points.

The Fluoroacetate team returned to attempt the 6th exploit of the competition, targeting Adobe Reader with a Windows local privilege escalation. They employed a a pair of use-after-free bugs - one in Acrobat and one in the Windows kernel - to elevate privilieges and to gain control over the system. That one earned them $50,000 and 5 points towards Master of Pwn.

The only exploit attempt to fail was the 7th and final attempt made by the Synacktiv team. They targeted VMware Workstation also in the Virtualization category. But, as I noted, the attempt failed when they were unable to get their exploit to function in the alloted time.

The Novel CoronaVirus

My recovery journey

Testing / The tests we have now:

- The (RT-PCR) Reverse transcription polymerase chain reaction.
- Fast to design but extremely labor-intensive, slow and expensive to run.
- Burns through PPE (Personal Protective Equipment)
- NEVER going to be practical as a solution for mass testing.
- Does **not** detect someone who had it and recovered.
- At best it's a useful emergency interim measure for use in routing critically ill patients.

What we need, and what's on the way, are known as ELISA tests. "ELISA" stands for "Enzyme-Linked ImmunoSorbent Assay." Here's a description of one of the very many pieces of work that's currently underway:

> To create the test, the researchers began by designing a slightly altered version of the "spike" protein on SARS-CoV-2's outer coat. (The alterations made the protein more stable for use in the lab.) That protein helps the virus enter cells, and it is a key target in the immune reaction against the virus, as the body churns out antibodies that recognize the protein and tag the virus for destruction. They also isolated the short piece of the spike protein called the receptor-binding domain (RBD), which the virus uses to attach to cells it tries to invade. They then used cell lines to produce large quantities of the altered spike proteins and RBDs.
>
> Those labmade molecules provided the basis for an ELISA test, in which antibodies in a sample of blood or plasma trigger a color change when they recognize a target protein—here an RBD or the spike protein. Initial tests of four blood samples from three confirmed COVID-19 patients and from 59 serum samples banked before the start of the outbreak showed that the test worked, as antibodies to SARS-CoV-2 bound to the test's proteins. It showed positive results only for the COVID-19 patients and not for any of those controls.

This second approach is the path to obtaining a "finger prick" test for COVID-19.

The other thing I wanted to comment upon is what we know, what we don't know, and the shape of the curve. In order to do any useful predictive modelling, having accurate data is everything. I am constantly infuriated when the popular press says "There are 30,000 cases of COVID-19 in the US." No, there aren't. 30,000 people who were tested were positive for COVID-19, but we have absolutely no idea whatsoever how many people in the US are positive for COVID-19. I may sound like I'm pedantic, but the distinction is crucial. Testing is the problem. It will always be a problem. Even when it gets much better, we'll still never have a true number of COVID-19 hosts, because there will always be people who didn't test.

But there is one number that we DO have, which, unfortunately, requires no testing: the number and rate of deaths due to COVID-19. It's unfortunate, but the death rate serves as a useful and accurate, though delayed, proxy for the total infection rate. It's delayed by two to three weeks. So, it is not very useful making short-term social distancing policy decisions. But it's brutal and accurate.

The reason I bring this up is the shape of the curve. In an environment like the one we're in, we

know NOTHING about the future until the moment the death rate curve peaks and begins to fall. It took China eight weeks to achieve. Italy, who is ahead of us on the curve, hasn't gotten there yet. And most unfortunately, this does assume a constant quality of care to hold the percentage of deaths constant. When the US's healthcase system is overwhelmed, as seems assured, the rate may accelerate for a time until the input to the medical system finally begins to wane.

Last week I shared the terrific ArsTechnica backgrounder on COVID-19, written by their inhouse PhD microbiologist. That URL was: https://grc.sc/covid.

This week I have two more wonderful discoveries to share. The first is for everyone in the household and I hope that after you've seen it you will arrange to put it up on a big screen to share with your family: https://grc.sc/covid2 It's an amazing animated presentation that explains so much about what we're all going though in simple terms. I'm astounded.

The second one will appeal to a far more rarified audience, since it's a medical-school level whiteboard explanation of where these coronaviruses have been originating, and in extremely wonderful detail -- for those who really want to know -- exactly what being infected with coronavirus does to the human body. https://grc.sc/covid3

I posted the link after I discovered it a few days ago, and someone named Ian in our grc.health newsgroup posted: "OK I am ready to move on to the anxiety counseling video now." It IS pretty intense. But if you're interested…

TRRespass

The fixes for RowHammer have not worked
We began covering RowHammer half this podcast ago, six years ago in 2014.

We should not confuse RowHammer with the more theoretical processor-architecture attacks such as Spectre and Meltdown. Spectre and Meltdown were important because they showed how the tricky designs used to increase processor performance could be leveraged against us. But they were never easy to pull off in the field. The researchers have shown that RowHammer is a much more real and significant threat.

To quickly recap, researchers discovered that the main bulk volatile DRAM lying at the heart of every system was not nearly as robust as we had always assumed. Over the years, under the pressure to deliver ever more DRAM density, the memory's storage cells had been successively reduced in size to the point where they were operating right on the hairy edge of "we really can't make them any smaller." And DRAM parity checking and error correction technologies, which were originally intended to protect against an impact by a stray cosmic ray, were increasingly being used to buffer the DRAM's underlying reduced reliability. We can think of this as a "noise margin," where there's a given certainty that a DRAM cell's voltage represents a 0 or a 1. And over time, in the pursuit of DRAM density, the noise margin was successively reduced.

The ever-clever researchers showed the world that by forcing atypical DRAM access patterns to deliberately create higher noise, it was possible to cause modern DRAM to malfunction in such a way that, with some control, individual bits could be flipped.

Since today's processors use DRAM-based tables to manage memory virtualization, they then demonstrated all of the various sorts of mischief that could be created by flipping bits in these DRAM-based management tables and much more. Malicious "RowHammering" processes could give themselves full access to other processes, read/write access to the OS kernel and more.

Through the ensuing years...

- Researchers showed how a Rowhammer attack could alter data stored in DDR3 and DDR4 memory.
- They showed how a Rowhammer attack could be carried out via JavaScript, via the web, and not requiring access to a PC, physically, or via local malware.
- They demoed a Rowhammer attack that took over Windows computers through the Microsoft Edge browser.
- They demoed a Rowhammer attack that took over Linux-based virtual machines installed in cloud hosting environments.
- They used a Rowhammer attack to get root permissions on an Android smartphone.
- They bypassed Rowhammer protections put in place after the disclosure of the first attacks.
- They showed how an attacker could improve the efficiency of a Rowhammer attack by relying on local GPU cards.
- They developed a technique to launch Rowhammer attacks via network packets.
- They developed a Rowhammer attack that targets an Android memory subsystem called ION, and which broke the isolation between the OS and local apps, allowing data theft and total device control.
- They developed a Rowhammer attack named ECCploit that works even against modern RAM cards that use error-correcting code (ECC)
- They discovered RAMBleed, a Rowhammer attack variation that can exfiltrate data from attacked systems, not just alter it.
- They developed a technique to speed up Rowhammer attacks with the help of field-programmable gate array (FPGA) cards, in an attack named JackHammer.

The solution to this RowHammer mess was supposed to be a series of mitigations collectively referred to as Target Row Refresh (TRR). And we've talked about that too. The idea being that noise immunity can be maintained if DRAM rows in the areas of unusually high activity are proactively brought up for refresh more often. TRR has been gradually implemented and rolling out for the past six years.

DDR4 memory cards were just coming online back then and so they were the first ones to receive the new TRR protections. And for while, vendors believed that they had finally plugged the Rowhammer issue. You do always have to worry a bit when you hear the term "mitigation" -- as in, well, we didn't really fix it but we made it much better.

The original RowHammer guys, led by Herbet Bos and his team from Amsterdam University in league with researchers from ETH Zurich and Qualcomm. Their recently released paper is titled: “TRRespass: Exploiting the Many Sides of Target Row Refresh”

In this research they outline their development of a generic tool named "TRRespass" that can be used to upgrade the old Rowhammer attacks to work on the new-and-improved TRR-protected RAM cards.

Here's now they summarized their work:
https://download.vusec.net/papers/trrespass_sp20.pdf

> After a plethora of high-profile RowHammer attacks, CPU and DRAM manufacturers scrambled to deliver what was meant to be the definitive hardware solution against the RowHammer problem: Target Row Refresh (TRR). A common belief among practitioners is that, on the latest generation of DDR4 systems that are protected by TRR, RowHammer is no longer an issue in practice. However, in reality, very little is known about TRR. How does it work? How is it deployed? And is it actually effective against RowHammer?
>
> In this paper, we demystify the inner workings of TRR and debunk its security guarantees. We show that what is advertised as a single mitigation is actually a series of different solutions coalesced under the umbrella term Target Row Refresh. We inspect and disclose, via a deep analysis, different existing TRR solutions, and demonstrate that modern implementations operate entirely inside DRAM chips. Despite the difficulties of analyzing in-DRAM mitigations, we describe novel techniques for gaining insights into the operation of these mitigations. These insights allow us to build TRRespass, a scalable blackbox RowHammer fuzzer that we evaluate on 42 recent DDR4 DIMMs.
>
> TRRespass shows that even the latest generation DDR4 systems with in-DRAM TRR, immune to all known RowHammer attacks, are often still vulnerable to new TRR-aware variants of RowHammer that we have developed. In particular, TRRespass finds that, on present-day DDR4 modules, RowHammer is still possible when many aggressor rows are used (even 19 in some cases), in a configuration we generally refer to as Many-sided RowHammer. Overall, our analysis shows that 13 out of the 42 DIMMs from all three major DRAM manufacturers (namely, Samsung, Micron and Hynix) are vulnerable to our TRR-aware RowHammer access patterns, and thus one can still mount existing state-of-the-art RowHammer attacks. In addition to DDR4, we also experiment with LPDDR4(X) chips and show that they are susceptible to RowHammer bit flips too. Our results provide concrete evidence that the pursuit of better mitigations must continue.

So, what has happened is that DRAM manufacturers looked at the existing RowHammer attacks. They didn't actually solve the RowHammer problem, because they really can't. The RowHammer problem does not arise from some defect in DRAM. It is insidious because it arises from the underlying technology of DRAM for which there can be no quick fix.

So what did they do? They designed internal, secret, proprietary, hardware mitigation workarounds for the various specific RowHammer attacks. So, the various things that were done before no longer work. The undaunted researchers reverse-engineered what was going on by very carefully examining the behavior of the updated DRAM. And once they had learned the tricks that had been incorporated they simply evolved different attacks to bypass those tricks.

- Observation 1: The TRR mitigation carries out a targeted refresh on every refresh command.
- Observation 2: The mitigation can sample more than one aggressor per refresh interval.

- Observation 3: The mitigation can refresh only a single victim within a refresh operation

Based on these observations, we conclude that hammering more than 4 rows should circumvent the mitigation. We confirm this by running a test on our FPGA infrastructure with standard conditions (i.e.,tREFI= 7.8µs).

Their conclusion adds a bit of surprising detail:

This paper shows that, despite significant mitigation efforts, modern DDR4 systems are still vulnerable to the RowHammer vulnerability—and even more vulnerable than before, once the mitigations are bypassed. In particular, we demonstrate that Target Row Refresh (TRR), publicized by CPU and DRAM vendors as the definitive solution to RowHammer, can be bypassed to cause RowHammer bit flips.

First, we show that TRR is an umbrella term for a variety of mitigations deployed at the memory controller or in DRAM chips. Second, we analyze common TRR implementations in the memory controller (using timing side channels) and in DRAM chips (using an FPGA-based DDR4 memory controller). Our analysis shows that in all our tested consumer platforms, TRR is only in effect inside DRAM chips. Futhermore, we discover that modern (in-DRAM) TRR implementations are generally vulnerable to many-sided RowHammer, a new hammering strategy that considers many aggressor rows in each hammering attempt. Finally, we present TRRespass, a blackbox many-sided RowHammer fuzzer that, unaware of the implementation of the memory controller or the DRAM chip, **can** still find sophisticated hammering patterns to mount real-world attacks for many of the DRAM modules in the market. Our results provide evidence that the pursuit of effective RowHammer mitigation must continue ***and that the security by obscurity strategy of DRAM vendors puts computing systems at risk for extended periods of time.***

DISCLOSURE

We disclosed our new RowHammer attacks to all affected parties in November of 2019. This triggered an industry-wide effort in addressing the issues raised in this paper. Unfortunately, due to the nature of these vulnerabilities, it will take a long time before effective mitigations will be in place. Further Developments on these vulnerabilities are tracked under CVE-2020-10255. The paper remained confidential until the public disclosure date of March 10, 2020.