Security Now! #867 - 04-19-22

A Critical Windows RPC RCE

This week on Security Now!

This week we examine Chrome's 3rd 0-day of the year, followed by Microsoft's massive 128-patchfest last week, and we note that we don't even bother counting Windows' 0-days, though there were another 2 this month amid the 47 CRITICAL vulnerabilities that were patched, one of them being so worrisome that it captured this week's podcast title which we'll cover at length before we conclude. We also have more WordPress add-on trouble, the return of a long standing problem in Apache Struts, and we have some interesting commentary about the current hackability status of the United States nuclear arsenal. I want to share a bit of closing the loop feedback with our listeners and give everyone a snapshot into the recent work on SpinRite. Then we're going to take a close look at the one flaw, out of 128 that Microsoft patched last week, that truly has the entire security industry on pins and needles — because it enables a zero-click Internet worm.

Where there's a will...

0-day Watch

Chrome's 3rd 0-day of 2022
Last Thursday, Chrome received an emergency update, bringing desktop Chrome to v100.0.4896.127 for Windows, Mac and Linux. The exploit, another type confusion flaw which was found in Chrome's V8 script processing engine, was found by Google's own Threat Analysis Group being exploited in the wild, thus the relative urgency to update. This is Chrome's third 0-day of the year, and its second type confusion 0-day. The first was a use-after-free in Chrome's animation support.

As usual, users of the Chromium-based browsers including Edge, Brave, Opera, and Vivaldi are similarly advised to look for any available updates to their web browser platforms.

There's no more to say about this. As usual, Google sees no benefit to telling us more than to please refresh our code. And this time I found that my instance of Chrome had already done so.

Patch Tuesday Redux

This is "Follow-Up Tuesday", our first podcast following Patch Tuesday, and this one won't disappoint. Users of Windows received a total of 128 fixes for known security vulnerabilities discovered in Windows desktop and server, Defender, Office, Exchange Server, Visual Studio, Print Spooler, Windows DNS server and more.

Of the 128 bugs which were addressed, 10 were rated Critical with a few even earning a rare 9.8 CVSS, which is, fortunately, especially rare for windows. We'll get back to those in a moment. A whopping 115 others are rated Important, and the final three are Moderate. Among those flaws are two 0-days, one that's known to be actively exploited in the wild and another that's been publicly disclosed.

The spread of problems breaks down as follows. Believe it or not, there were a total of 47 remote code execution vulnerabilities repaired. 47! And the same number of elevation of privilege vulnerabilities were fixed. 47 — each. Unbelievable. Lest anyone thought that the troubles with Windows Print Spooler were all behind us, 15 of those 47 privilege escalation vulnerabilities were found in Print Spooler. There were also 13 information disclosure problems fixed, 9 denial of service and 3 spoofing vulnerabilities. And Edge also separately had 26 flaws fixed.

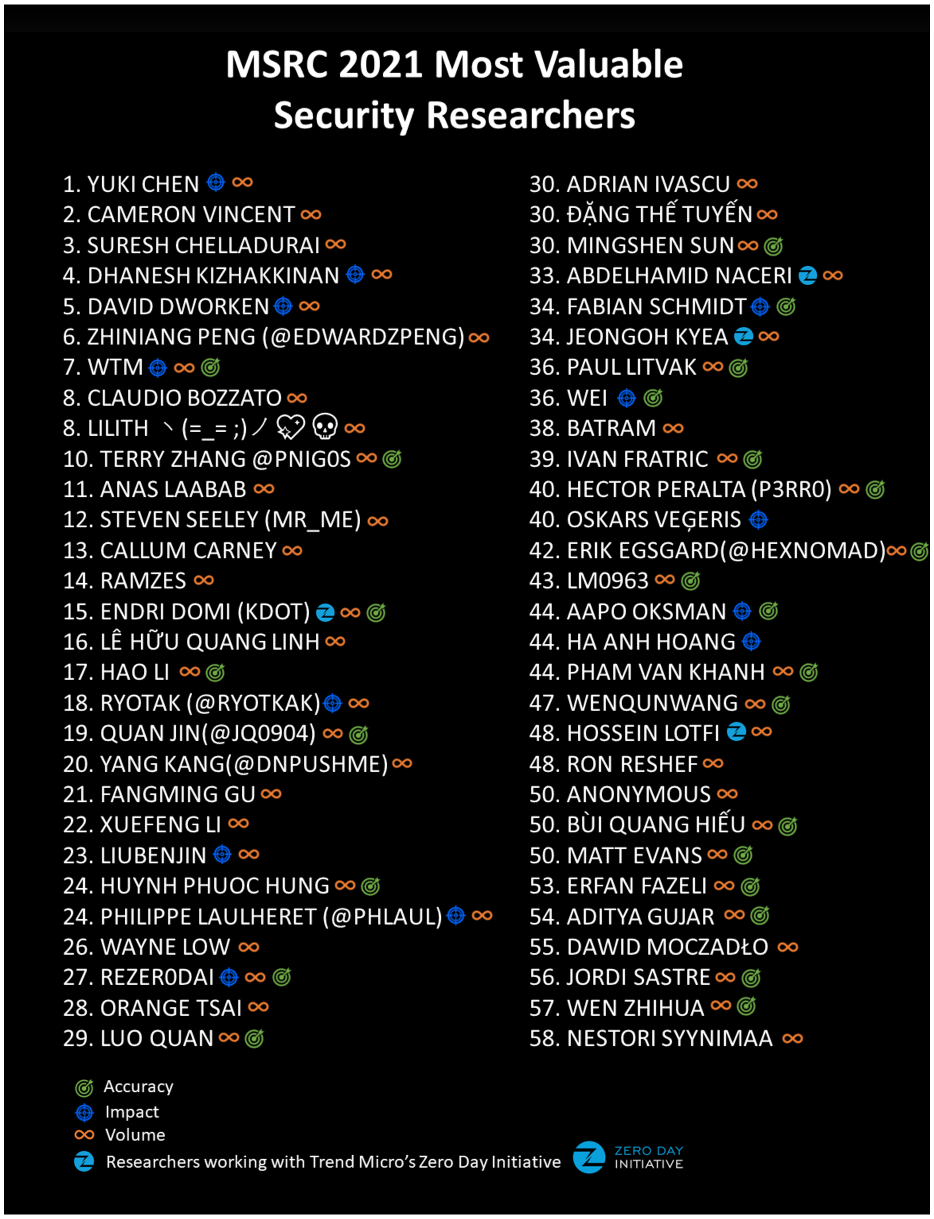
The flaw that's being actively exploited is an elevation of privilege vulnerability occurring in the Windows Common Log File System. It's been rated at a CVSS of 7.8. And interestingly, it was found and reported by both the U.S. National Security Agency (NSA) and some researchers at CrowdStrike.

The other 0-day which has been publicly disclosed but is not known to be under active exploitation, most likely because it's difficult to leverage, is also a privilege escalation, this one occurring in the Windows User Profile Service. Since its successful exploitation requires an attacker to win a race condition where timing is critical, it carries a somewhat lower CVSS of 7.0.

However, there are two biggies among the crop, both coming in with rare-for-Windows CVSS's of 9.8! The least worrisome of the two is in Windows' Network File System. And the other 9.8'er that has the security industry a bit on edge (we'll talk more about in a minute) is a remote code execution flaw in the often exposed Windows Remote Procedure Call (RPC) Runtime Library. Other critical RCE flaws were fixed in Windows Server, Windows SMB, and Microsoft Dynamics 365.

Microsoft's DNS server apparently came under well-known security researcher Yuki Chen's talented gaze, since Microsoft credited Yuki with discovering and responsibly disclosing a total of 18 previously unknown problems in their DNS server, one being an information disclosure flaw, but the other 17 being remote code execution flaws! 17 RCE's in their DNS server! Unbelievable.

I mentioned that Yuki was well known, and especially to Microsoft. Last summer Yuki tweeted: "Made #1 this year :) Thanks to @msftsecresponse and the bounty team. Congrats to all researchers on the list. I personally know some researchers who reported nice bugs but are not on the list this year, hat tip to their great work too. 4:00 PM · Aug 4, 2021·Twitter Web App


MSRC 2021 Most Valuable Security Researchers

1. YUKI CHEN ⊕ ∞
2. CAMERON VINCENT ∞
3. SURESH CHELLADURAI ∞
4. DHANESH KIZHAKKINAN ⊕ ∞
5. DAVID DWORKEN ⊕ ∞
6. ZHINIANG PENG (@EDWARDZPENG) ∞
7. WTM ⊕ ∞ ◎
8. CLAUDIO BOZZATO ∞
8. LILITH ヽ(=_=;)ノ 💔 💀 ∞
10. TERRY ZHANG @PNIG0S ∞ ◎
11. ANAS LAABAB ∞
12. STEVEN SEELEY (MR_ME) ∞
13. CALLUM CARNEY ∞
14. RAMZES ∞
15. ENDRI DOMI (KDOT) Ⓩ ∞ ◎
16. LÊ HỮU QUANG LINH ∞
17. HAO LI ∞ ◎
18. RYOTAK (@RYOTKAK) ⊕ ∞
19. QUAN JIN(@JQ0904) ∞ ◎
20. YANG KANG(@DNPUSHME) ∞
21. FANGMING GU ∞
22. XUEFENG LI ∞
23. LIUBENJIN ⊕ ∞
24. HUYNH PHUOC HUNG ∞ ◎
24. PHILIPPE LAULHERET (@PHLAUL) ⊕ ∞
26. WAYNE LOW ∞
27. REZER0DAI ⊕ ∞ ◎
28. ORANGE TSAI ∞
29. LUO QUAN ∞ ◎
30. ADRIAN IVASCU ∞
30. ĐẶNG THẾ TUYẾN ∞
30. MINGSHEN SUN ∞ ◎
33. ABDELHAMID NACERI Ⓩ ∞
34. FABIAN SCHMIDT ⊕ ◎
34. JEONGOH KYEA Ⓩ ∞
36. PAUL LITVAK ∞ ◎
36. WEI ⊕ ◎
38. BATRAM ∞
39. IVAN FRATRIC ∞ ◎
40. HECTOR PERALTA (P3RR0) ∞ ◎
40. OSKARS VEĢERIS ⊕
42. ERIK EGSGARD(@HEXNOMAD) ∞ ◎
43. LM0963 ∞ ◎
44. AAPO OKSMAN ⊕ ◎
44. HA ANH HOANG ⊕
44. PHAM VAN KHANH ∞ ◎
47. WENQUNWANG ∞ ◎
48. HOSSEIN LOTFI Ⓩ ∞
48. RON RESHEF ∞
50. ANONYMOUS ∞
50. BÙI QUANG HIẾU ∞ ◎
50. MATT EVANS ∞ ◎
53. ERFAN FAZELI ∞ ◎
54. ADITYA GUJAR ∞ ◎
55. DAWID MOCZADŁO ∞
56. JORDI SASTRE ∞ ◎
57. WEN ZHIHUA ∞ ◎
58. NESTORI SYYNIMAA ∞

◎ Accuracy
⊕ Impact
∞ Volume
Ⓩ Researchers working with Trend Micro's Zero Day Initiative

ZERO DAY INITIATIVE

Security News

WordPress once again…

A site named "Plugin Vulnerabilities" explains itself as "A service to protect your site against vulnerabilities in WordPress plugins." (That ought to keep them busy.) Their recent posting from last week is titled: *"5+ Million Install WordPress Plugin Elementor Contains Authenticated Remote Code Execution (RCE) Vulnerability"*

Late last week, third-party data we monitor showed what was possibly a hacker probing for usage of a WordPress plugin Elementor, which has 5+ million active installs according to WordPress, by the requesting this file:

```
/wp-content/plugins/elementor/readme.txt
```

We couldn't find any recent disclosed vulnerabilities that should explain that, so we started doing our standard checks we do in a situation where a hacker may be exploiting an unfixed vulnerability in a plugin. What we immediately found was that the plugin isn't handling basic security correctly, as we found many functionalities where capabilities checks were missing where they shouldn't. While some of those were not accessible to users that shouldn't have access, we found at least one that is and the functionality accessible leads to one of the most serious types of vulnerabilities, remote code execution (RCE). That means that malicious code provided by the attacker can be run by the website.

In this instance, it is possible that the vulnerability might be exploitable by someone not logged in to WordPress, but it can easily be exploited by anyone logged in to WordPress who has access to the WordPress admin dashboard. Unless another plugin restricts access to the admin dashboard, that would mean anyone logged in to WordPress would have access.

The vulnerability was introduced in the plugin in version 3.6.0, which was released on March 22. According to WordPress' latest stats, 30.3 percent of users of the plugin are now on version 3.6.x.

Based on just what we saw in our very limited checking, we would recommend not using this plugin until it has had a thorough security review and all issues are addressed. That it has 5+ million installs and hasn't been properly secured should be very concerning. It certainly isn't for a lack of money at the developer, as they raised 15 million dollars in 2020. It also isn't for a lack of reason to be concerned, as two years ago it was claimed a zero-day vulnerability in paid version of the plugin was being exploited.

Their posting then shows snippets of code to demonstrate and detail the vulnerability. And apparently this disclosure, their disclosure, was made deliberately and irresponsibly over a dispute with the WordPress Plug-In forum moderators. It seems that these "Plugin Vulnerabilities" guys have long been unhappy with the way WordPress manages the reporting of security vulnerabilities. Reading between the lines, it sounds as though WordPress downplays vulnerability reports which annoys these "Plugin Vulnerability" guys. Sounds like a clash of egos.

A different firm known as "PatchStack" also addressed this issue. They wrote "The widely popular WordPress website builder plugin Elementor, which has over 5 million active installations, has recently released version 3.6.3 which contains an important security fix. This vulnerability could allow any authenticated user, regardless of their authorization, to upload arbitrary files to the site. The arbitrary file upload vulnerability could allow someone to take over the entire site or perform remote code execution (RCE). Please update immediately!”

Then they added that “Patchstack PRO and Business users have received a virtual patch to be protected from this vulnerability.” and the previous group with their bruised egos also produce what they call a WordPress Firewall to autonomously protect their subscribers from this danger — when it’s been configured to do so — in this case to limit the types of files which can be uploaded.

So, WordPress’s undeniable popularity, coupled with the constant stream of problems, mostly created by insecure and poorly written Wordpress plugins, has spawned an industry of add-on WordPress protectors. In the past we’ve often mentioned the firm “Wordfence”. We just talked about two others, “Plugin Vulnerabilities” and “PatchStack”... and there are many others.

Although it was obviously self-serving, recall that it was Patchstack who, earlier last month, and we covered it at the time, released a research whitepaper observing that last year saw a 150% increase in reported WordPress vulnerabilities compared to 2020, with the alarming news that 29% of the critical flaws in WordPress plugins never received a security update. They also observed that only 0.58% (so, about 1 in 200) were found in the WordPress core, with the rest being on themes and plugins written by anyone else and offered without review. They also noted that almost all of the problems — 91.38% of the flaws — were found in free plugins, whereas paid/premium WordPress add-ons only accounted for the other 8.62% of the total.

So I’ll repeat the best advice, which would be to stick with the most bare bones Wordpress installation possible. If you want more, pay for it from a reputable WordPress add-on provider. And accept that, as always, there will be some tradeoff of ease of use for security to decide where you want to be.

Apache Struts Framework needs a critical update

When we hear the phrase “Apache Struts” our mind immediately jumps to the historic Equifax breach on 2017 which was the result of them not having updated their Apache Struts Java web framework for several months after a critical “must patch” update had been published. WIRED Magazine’s coverage of the event was titled “Equifax Officially Has No Excuse” with the subtitle “A patch that would have prevented the devastating Equifax breach had been available for months.” That infamous breach compromised the data of 143 million users as hackers exfiltrated the names, Social Security Numbers (SSNs), dates of birth, addresses, and, in some cases, driver's license numbers.

What wasn't mentioned at the time was that the flaw was in an historically troubled Struts component known as OGNL — Object-Graph Navigation Language. And guess what, it's a powerful interpreter. OGNL is an open-source Expression Language (EL) for Java which simplifies the range of expressions used in the Java language. Among other things, OGNL enables coders to more easily manipulate arrays. But, as it turns out, parsing OGNL expressions based on untrusted or raw user input has long been dangerous. Listen to what Wikipedia has to say, and don’t try to understand it. Just let it sort of wash over you...

OGNL began as a way to map associations between front-end components and back-end objects using property names. As these associations gathered more features, Drew Davidson created Key-Value Coding language (KVCL). Luke Blanshard then reimplemented KVCL using ANTLR and started using the name OGNL. The technology was again reimplemented using the Java Compiler Compiler (JavaCC).

OGNL uses Java reflection and introspection to address the Object Graph of the runtime application. This allows the program to change behavior based on the state of the object graph instead of relying on compile time settings. It also allows changes to the object graph.

Due to its ability to create or change executable code, OGNL is capable of introducing critical security flaws to any framework that uses it. Multiple Apache Struts 2 versions have been vulnerable to OGNL security flaws. As of October 2017, the recommended version of Struts 2 is 2.5.13. Users are urged to upgrade to the latest version, as older revisions have documented security vulnerabilities — for example, Struts 2 versions 2.3.5 through 2.3.31, and 2.5 through 2.5.10, allow remote attackers to execute arbitrary code. Atlassian Confluence has been affected by an OGNL security issue that allowed arbitrary remote code execution, and required all users to update.

In other words, as we’ve seen before, not all ideas are good, and OGNL appears to be WAY too powerful and also particularly gifted at being bad. Wherever it’s used it appears to wreak havoc.

Back in 2020 a different “OGNL Injection Bug”, which is what all of these OGNL problems are called, was assigned CVE-2020-17530 with an attention grabbing severity rating of 9.8 (Critical). Back then, a pair of researchers responsibly reported to the Struts team their discovery of a "double evaluation" flaw in Struts2 versions 2.0.0 - 2.5.25, which could occur under certain circumstances. The advisory for that CVE states that “Some of the tag's attributes could perform a double evaluation if a developer applied forced OGNL evaluation by using the %{...} syntax.” and “Using forced OGNL evaluation on untrusted user input can lead to a Remote Code Execution and security degradation.”

Although Apache had resolved the known 2020 flaw in the succeeding release, Struts 2.5.26, researcher Chris McCown later discovered that the applied fix was incomplete. So he responsibly reported to Apache that the so-called “double evaluation” problem could still be reproduced in Struts versions 2.5.26 and above which resulted in the assignment of last year's CVE-2021-31805. Consequently, users are advised to immediately upgrade to Struts 2.5.30 or greater and to avoid using forced OGNL evaluation in the tag's attributes based on untrusted user input. It's just not safe. And given what happened after Equifax apparently ignored similar advice back in 2017, I’d be inclined to do whatever was necessary to stay current. If you're using Struts 2, be sure that you're running the latest, which will now be at least 2.5.30.

Are America’s nuclear systems so old they’re un-hackable?

Back in 1984, Judy, a dear friend of mine who was incredibly non-computer savvy, was a realtor.

We met when I purchased my home from her in 1984. She needed to access the realtor's MLS, multiple listing service, and she was using something she called her "modem" which was actually a Texas Instruments Silent 700 thermal printing terminal with those original rubber cups on top that you would press a telephone's handset into after dialing into a computer.

We became lifelong great friends. When the Internet happened, the MLS service moved online. So I set her up with a Windows 95 machine. Many years went by with everything working fine, until someone she had over to her home was shocked that she was still using Windows 95. I don't recall where the rest of the world was by then. Probably at least on WindowsXP. So she immediately phoned to ask why her computer had been allowed to become obsolete. I said: "Judy... Does it work?" Which gave her pause. She said, "Uh, yes." I asked: "Is there anything you need it to do that it doesn't?" and she said: "No." So I explained that there was a growing problem on the Internet with security. People were getting their computers hacked by clicking on the wrong thing. She said: "Oh yeah! That's happened to several of my friends. They got viruses in their computers that were sending out eMails to all of their friends and infecting them, too." So I said: "Judy, did you receive those eMails?" She said "Yes." I asked: "And did you get infected too?" and she said "No." And I said: "That's right. And that's because your computer is too old to get infected by modern viruses. It uses an older and original sort of code, which is in many ways better, especially for the few things you need your computer to do." She continued to use that machine happily for many more years until someone convinced her to buy a new one, after which she pretty quickly got herself infected.

And that brings us to a really interesting piece that ran in "The Record" last week that I wanted to share some parts from. What was intriguing to me, and I think will be intriguing to our listeners, was how much of the general philosophy of complexity and security, and the "if it's not broke don't fix it" philosophy we've developed on this podcast through the years, appeared in The Record's quotes.

To set up their piece, The Record first establishes the context of the moment. They wrote:

As the Cold War drew to a close a surprising contender emerged as the third largest nuclear power on earth: Ukraine. The country was home to some 5,000 nuclear weapons, placed there by Moscow when Ukraine was still part of the Soviet Union. Kyiv sent the weapons back to Russia in exchange for security guarantees from the U.S. and Britain and a promise from Moscow that it would respect Ukraine's its sovereignty.

Then, President Vladimir Putin invaded in February.

The nuclear option, which many thought had been largely removed from the table, was one of the first sabers Putin chose to rattle when he announced that Russian troops were moving into Ukraine in February. He reminded the world that not only did Russia possess nuclear weapons, but it was prepared to use them. Anyone who "tries to stand in our way," he said, will face consequences "such as you have never seen in your entire history."

The threat raised an uncomfortable question: After decades of pursuing disarmament talks and assuming nuclear confrontation was a bridge too far, was the United States ready for the ultimate confrontation with Russia?

Okay. That's interesting. Amid all of the talks of cyber offense and defense I've been wondering, as I imagine our listeners may have, how strong our nuclear deterrent is in the face of reportedly quite capable foreign cyber attacking adversaries?

What I'm going to share from what The Record wrote, which I've edited for the podcast, speaks to exactly that:

Right up until three years ago, U.S. nuclear systems were using eight-inch floppy disks in a IBM System 1 computer first introduced in 1976. It was not connected to the internet and required spare parts often sourced from eBay. Some analysts think America's slow-walk toward modernization of its nuclear systems may turn out to have been a canny strategy: because the systems are so old, they are practically un-hackable.

Herb Lin, a professor at Stanford University and author of a new book titled "Cyber Threats and Nuclear Weapons" said: "There is a truism about computers, which is that when we have a computer, we always want it to do more. " His book looks at the risk of cyber attacks across the entire nuclear enterprise. He says the "more" problem inevitably introduces vulnerability into the system, and defense officials have to think carefully about how to modernize.

He said: "If you'll grant the point that the more you want a computer system to do, the more complex the system is that you have to build, then you take the second step and you realize that complexity is the enemy of security." [Gee, I wonder where we've heard that before?]

He says that this is where things start to go wrong. Run the probabilities and there's a chance that one of those many complex components could be vulnerable to a hack in a way no one had considered before.

The cautionary tale is Stuxnet, the virus and worm that found its way into the Natanz uranium enrichment plant in Iran in 2009 and 2010. Stuxnet – which appears to have been the brainchild of U.S. and Israeli intelligence services – was able to take control of centrifuges used to enrich uranium gas inside the plant and, without anyone noticing, get them to spin so fast they broke down.

For a long time, the cause of the centrifuge failures was a complete mystery. Scientists were fired – officials thought they were sleeping on the job or not maintaining the systems properly. It never occurred to anyone – until much later – that a cyber weapon could possibly find its way into a system that was air-gapped from the Internet and so closely watched. Stuxnet had probably been in their systems a year before they even discovered it.

The thinking has been that America's geriatric nuclear weapons systems may actually provide an inoculation from this kind of attack. Lin wrote: "Many of the systems right now are so old that there's nobody or few, very few, people who know how to get at them. So right now the current assessment is that the nuclear command and control system anyway…is mostly robust against a cyber threat."

Hayat Alvi, a professor at the U.S. Naval War College, studies these kinds of nuclear weapons issues. (She spoke to The Record in her personal capacity.) She says she has a mantra when it comes to our nuclear weapons systems: "If it's not broken, it doesn't need to be fixed."

Alvi says the calculus is pretty simple: "Why try to change something that has worked for decades? Assuming you change them to upgrade them to modern technology, you are actually inviting more risks and potential threats and sabotage into the system."

While officials have tinkered at the edges of the nuclear weapons systems, John Lauder, who used to direct the CIA's Nonproliferation Center, says "most of the systems we're using are from the '70s and '80s."

He said there has been a general sense from people who worked in arms control that "we had put together a set of agreements that would keep peace and stability." As a result, modernizing nuclear weapons systems seemed less important since there was a general sense the weapons would eventually be phased out. "Ukraine," he said, "was a wake-up call."

('One horrifying moment…')

In 1979, about three years after the U.S. nuclear weapons program adopted the state-of-the-art IBM Series/1 computer, William Perry, who was a top Pentagon official at the time, got a phone call. The voice on the other end identified himself as the watch officer.

Perry, who would later go on to be Defense Secretary in the Clinton administration, recounted in his podcast, "At the Brink": "The first thing he said to me was that his computers were showing 200 nuclear missiles on the way from the Soviet Union to the United States. And for one horrifying moment, I believed we were about to witness the end of civilization."

As Perry weighed the possibilities, he concluded this had to be some kind of mistake. There was nothing going on in the world at the time that would have caused the Soviet Union to suddenly strike. Perry asked the watch commander to find out what had gone wrong with the systems so he could explain what happened to the president in the morning.

It turns out, someone had accidentally put a training tape into the computer instead of an operating one. As a result, what the computer saw was a simulation of an actual attack. It looked real because it was designed to look real. Perry says that night fundamentally changed the way he thought about nuclear weapons. He came to the conclusion that simple human error could indeed lead to nuclear war.

He said: "It has changed forever my way of thinking about nuclear weapons. Up until this, a false alarm, an attack by mistake, starting a nuclear war by mistake was a theoretical issue."

Until it wasn't.

There have been many questions raised and interesting movies made surrounding the question of whether a human being would be able to follow an order to turn the keys to launch a strike.

So there’s a huge urge on the part of those who are given the responsibility of protecting us to remove the human factor from the loop on the basis that it introduces a wildcard — an unknowable uncertainty that cannot be relied upon. The good news is that so very many lessons have since been learned about the true fragility of our supposedly advanced technology that it’s at least reasonable to hope that ultimate control will not be centralized. After all, the strength of the Internet arises from its inherently and brilliantly decentralized design.

Closing The Loop

Max Feinleib / @MaxFeinleib

“Re the Coinbase segment: Gmail also has the same login process where you enter your email, and then your password on another screen.”

Right. Many other listeners have written to note that this or that online service does the same thing as Coinbase. I distinctly recall that once upon a time we used gmail as an example of this being done right. But, sure enough, I tested gmail with the “bingo-zonk-dingo-lingo” user and I never had the chance to enter a login password. Gmail immediately told me that they had no record of that account. So it appears that ease-of-use have prevailed, even at Google, over stricter but less user-friendly and less forgiving login policies.

Dana J. Dawson / @djdawso

“@SGgrc On SN 866 you seemed unimpressed with the new MS Windows auto update service, but in other SN episodes you've implored home router makers to include an auto update feature. Could you elaborate on this apparent contradiction on an upcoming SN episode?”

The issue of auto-upgrades isn’t settled because it’s not black and white. There are pros and cons and tradeoffs, and there isn’t a single right answer. The biggest difference arises due to the environment being addressed. For example, a complex enterprise network has many moving pieces and it presumably has some dedicated IT staff who know those pieces and are competent to test any changes and validate that they are safe before they are applied to the entire network.

Contrast that to a little router box, sitting alone and forgotten in a closet, which was installed by an elderly couple’s grandson after they asked him how they could get WiFi. If, when it was installed, the little router offered the option for automatically keeping itself up to date, we can hope that the grandson had the foresight to enable it if that wasn’t its default. My feeling is that given the “set and forget” environment which most such routers find themselves in, coupled with the fact that those routers are bristling with features and run complex Linux operating systems where new problems are being found, and that there’s active pressure to find and compromise these autonomous little devices for incorporation into botnets, cryptomining and perhaps someday something more, I believe that the default setting should be to allow the router to repair itself in the event that a critical remotely exploitable flaw is discovered and reported to its manufacturer.

The enterprise and the router-in-the-closet scenarios are clearly very different. But what they have in common is their use of complex software which we still haven't figured out how to get right. I think that the use-case for AutoPatch is the very large middle ground. I'm unconvinced that it will make sense for any significant enterprise that employs their own in-house IT staff to turn over control of updating to Microsoft. But most small businesses do not have in-house IT staff. They have some guy who comes in when they call to fix something that broke. If Microsoft has found that their monthly updating is breaking more things than it's fixing, then having a way to detect when something has gone wrong and revert — in other words, to "Auto Patch" — makes sense. And "Auto Patch" sounds more appealing than "Windows Un-Update."

To me, it still feels eerily like capitulation. As if somewhere inside Microsoft someone acknowledged that they cannot actually fix Windows, so they're going to automate it not being fixed. But I still love Windows. I think it's an amazing piece of work. It's in front of me right now. I think it's incredible that it works as well as it does, and I very rarely have any trouble with it. And that's clearly the case for most of the world.

SpinRite

Though I haven't mentioned SpinRite here for a few weeks, work has continued without interruption. We've just emerged from an interesting sideline where we learned some surprising things about many of our systems. One of the big changes being made in SpinRite v6.1 is the use of very large 16 megabyte transfer buffers. The theory was that this would allow maximum performance by reducing inter-transfer overhead. And for SSD's this theory has panned out. It also holds for some spinning drives, which we refer to as "spinners." But it turns out that not **all** spinning drives perform best when they are handed huge transfers to perform. I don't now recall why I thought to look, but we were finding that some BIOSes were outperforming SpinRite's own native drivers, which should not have been possible. So I added technology to benchmark drives both ways — using SpinRite's new native hardware-level drivers and using the BIOS — with the intention to use the BIOS when it was faster than SpinRite. The BIOS has never been able to transfer more than 127 sectors at a time. But SpinRite can ask drives to transfer 32,768 sectors at once into a 16 megabyte buffer. Through experimentation, we discovered that some spinners would perform better when asked for 258 individual transfers of 127 sectors each rather than a single transfer of 32,768 sectors. But my code was still not keeping up with the BIOS. Then I realized that the way I was breaking up the large transfer into smaller transfers was introducing 258 times the inter-transfer overhead. So I redesigned my drivers to be maximally efficient when larger transfers needed to be divided into smaller pieces. And then, finally, SpinRite's own drivers were always at least as fast as the BIOS and often, especially for SSDs, up to several times faster. So SpinRite will never need to use the BIOS when it's able to communicate with drive hardware directly. SpinRite now measures the speed of each drive with 127-sector transfers and 32,768-sector transfers and remembers which one is best for that drive and adapter.

I know this is all taking time. The only thing I can say is that what is emerging is going to be a truly high functioning piece of software — something that anyone can be proud to own and use. And it's also creating a strong new foundation for SpinRite's future beyond v6.1.

A Critical Windows RPC RCE

CVE-2022-26809 with a CVSS of 9.8.

Last Tuesday, when, along with 127 other lesser problems, Microsoft patched this flaw in Windows' remote procedure call runtime library DLL (rpcrt4.dll), it was unknown to anyone other than Microsoft and its independent discoverer, "BugHunter010" with the Chinese KunLun Lab, who had privately and responsibly disclosed it to Microsoft. As the implications of this freshly patched flaw began to emerge the entire security industry started sitting up straighter in their chairs, because what we have here is something rare: In any not-yet-patched Internet-exposed Windows machines we have the basis for another zero-click Internet worm, reminiscent of 2003's MS Blast worm and the 2017 Wannacry attack.

CERT/CC's Will Dormann tweeted that right now, 1,329,075 Windows machines are publicly reachable and until patched, are vulnerable. And all of those machines which remain unpatched once the discovery is made of how to exploit this vulnerability will be susceptible to full unauthenticated remote takeover.

Microsoft's page disclosing and summarizing this vulnerability categorizes as:

- Attack Vector: Network
- Attack Complexity: Low
- Privileges Required: None
- User Interaction: None
- Confidentiality Impact: High
- Integrity Impact: High
- Availability: High
- Exploitation: More Likely

In other words, Microsoft is quite aware that it doesn't get any worse than this. And at this very moment everyone, and I mean everyone, is working around the clock to figure out how to turn this into an active exploit. Security companies are trying to do it in order to build defenses for their clients, and we **know** that all of the usual suspects in North Korea, China and Russia are burning the midnight oil to design packets that they can send to any listening server to take it over remotely.

We all know that when we start off with more than 1.3 million vulnerable Windows machines, a month from now, somewhere between 10% and 25% will still be vulnerable, and it will be their misfortune. A working exploit for this would be extremely valuable and would likely fetch a high price — even though a patch exists — since everyone knows that merely reduces the size of, but does not eliminate, the target population. So if an exploit were developed it would be kept quiet since anyone selling it would want it to be rare, and anyone paying a high price would want to keep it for themselves or for their own selective resale. So, I wouldn't expect to see a worm appear initially. Deploying this bad boy in targeted attacks makes a lot more sense.

Last Wednesday, the day after last week's patches became available, Akamai's security people, seeing how Microsoft had characterized this and seeing its 9.8 CVSS, became curious about this change that Microsoft had made and they posted a blog about the vulnerability because it's very likely going to become famous... or infamous. I've excerpted and edited from their blog to give everyone a sense for what Akamai did...

The CVE stated that the vulnerabilities lie within the Windows RPC runtime, which is implemented in a library named rpcrt4.dll. This runtime library is loaded into both client and server processes utilizing the RPC protocol for communication.

We compared versions 10.0.22000.434 (unpatched, from March 2022) and 10.0.22000.613 (patched, from April 2022) to produce a list of changes in various functions. The functions ProcessResponse and ProcessReceivedPDU caught our eye. The two functions are similar in nature; both process RPC packets, but one runs on the client side and the other on the server side. We went on to "diff" ProcessReceivedPDU, and noticed two code blocks that were added to the new version.

In the patched code, we saw that a new function was called after QUEUE::PutOnQueue. Zooming in on the new function and inspecting its code, we saw that it checks for integer overflows. Namely, the new function was added to verify that an integer variable remained within an expected value range after having another value added to it.

Diving deeper into the vulnerable code in "GetCoalescedBuffer", we noticed that the integer overflow bug could lead to a heap buffer overflow, where data is copied onto a buffer that is too small to populate it. This, in turn, allows data to be written out of the buffer's bounds, onto the heap. When exploited properly, this primitive could lead to remote code execution. A similar new call to check for integer overflow was added to three other functions as well. The integer overflow vulnerability and the function that prevents it, exists in both client-side and server-side execution flows.

Marcus Hutchins has posted a seven and a quarter minute video on YouTube titled "Exploiting Windows RPC - CVE-2022-26809 Explained | Patch Analysis"
https://www.youtube.com/watch?v=GGlwy3_jVYE

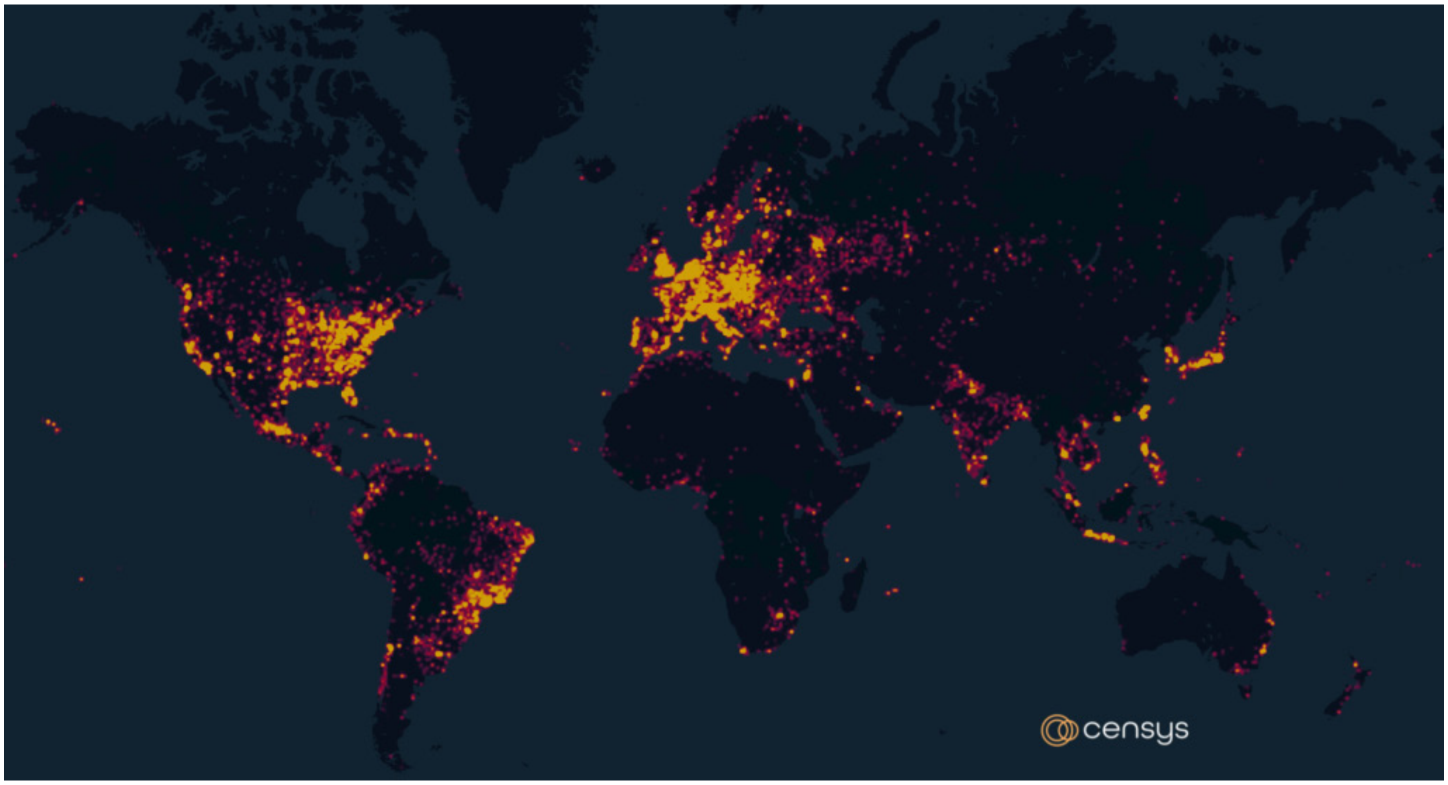
And the SANS Institute has posted an hour long YouTube video titled:
"CVE-2022-26809 MS-RPC Vulnerability Analysis - SANS Institute"
https://www.youtube.com/watch?v=fQ06VUq3kd8

In the video's description, SANS introduced the topic by writing: *"On Tuesday, April 12th, Microsoft released patches for CVE-2022-26809, reportedly a zero-click exploit targeting Microsoft RPC services. At the time of the publication of this abstract, there is no proof of concept available in the wild. However, based on the rating that exploitation is "more likely" we expect this won't last long."* And I suspect they are correct.

I've gotten this far into the subject without having made any mention of ports. Part of the reason is that RPC is not constrained to any single port, though it is typically carried over several, and one in particular.

Believe it or not, the #1 culprit is SMB port 445 which is used by Windows infamous file and printer sharing. Everyone who's been listening to this podcast for any length of time will have heard me lament, many times depending upon how long you've been listening, that Microsoft's security for their Internet services has never been worthy of trust. I mean, come on, their DNS server **just** had 17 remote code execution flaws repaired. 17! Just last week. The troubles with their remote desktop server have been numerous, and file and printer sharing has never been trustworthy.

The research company Censys said 1,304,288 hosts are running the SMB protocol as of last Wednesday. Of those, 824,011 (63%) were identified by Censys as running a Windows-based operating system — meaning that every one of those 824,011 machines was vulnerable. And Censys noted that they were unable to determine the OS behind approximately 369,485 (28%) hosts running SMB. The majority of the systems running SMB are in the United States with a count of 366,000 systems. Russia came in with a count of 144,622, Hong Kong with 72,885, 70,980 in Germany and 56,659 in France.



Last Wednesday the 13th, CERT's Will Dormann tweeted: *"CVE-2022-26809 Yes, blocking 445 at your network perimeter is necessary but not sufficient to help prevent exploitation. If by April 2022 you STILL have SMB exposed to the broader Internet you've got some soul searching to do. Now, about those hosts already inside your network..."*

And the day after that, SANS tweeted: *"Please remember: Port 445 is just ONE of the ports that may reach #RPC (CVE-2022-26809) on Windows. #MSRPC does Port 135 or in some cases HTTP as well."*

And another researcher, Ned Pyle tweeted: *"CVE-2022-26809 has nothing to do with SMB, it's an*

RPC vuln where a variety of transports can be used, like TCP/135, SMB/445, etc."

No matter how much you want to. No matter how much you may need to, other than HTTP and HTTPS, and I suppose Exchange Server is necessary too, other Windows services and their ports **must not be exposed** to the public Internet. This is why site-to-site packet filtering by IP address and port is crucial, and is a perfect solution when IPs are fixed, or even relatively fixed, and some other solution such as a VPN, SSH or Port Knocking needs to be used when a roaming IP must be able to obtain access.

My choice at this point would be to run a Wireguard VPN. We've talked about Wireguard before. It's a lovely, recently written, state of the art, 100% free, open source VPN which exchanges certificates for authentication and, unlike OpenVPN, deliberately leaves the kitchen sink where it belongs, in the kitchen. It does just one thing and it does it well. And there are clients for all platforms. So connecting to it is no problem. Although it was originally built on Linux, Windows is now officially supported. So you could run a Wireguard VPN server on any Linux, Mac or Windows machine on your network to which any number of roaming remote users could securely connect to obtain access to everything inside.

https://www.wireguard.com/install/
https://www.henrychang.ca/how-to-setup-wireguard-vpn-server-on-windows/
https://www.smarthomebeginner.com/wireguard-windows-setup/

There are now some How-To Setup Wireguard VPN on Windows pages and as I noted, Windows is officially supported.

But for those 1.3+ million systems which CERT says were vulnerable last Tuesday, it's going to be interesting to see what we're talking about in the next few weeks to come.

Before we wrap up this topic and podcast, I need to mention what many others have also noted, which is that the threat from this unpatched RPC RCE exploit is not only external. There is no chance that bad guys who gain entry into an enterprise's network will not be adding a test for this patch to their lateral movement toolkits.

Remember that many initial network penetrations are with low and limited privilege. This is the local operating system trying to protect itself. This is why privilege escalation or elevation exploits are almost as highly prized as remote code execution exploits. This RPC RCE will doubtless be added to all post-penetration kits because it would allow any unprivileged attacker who has obtained a minimal foothold inside a network to hugely expand their reach by moving laterally to other more valuable machines while simultaneously elevating their privileges on that destination machine to SYSTEM level.

In other words, as those tweets were reminding, this is not just about preventing external public exposure to those services. The threat from internal abuse is very real, too.