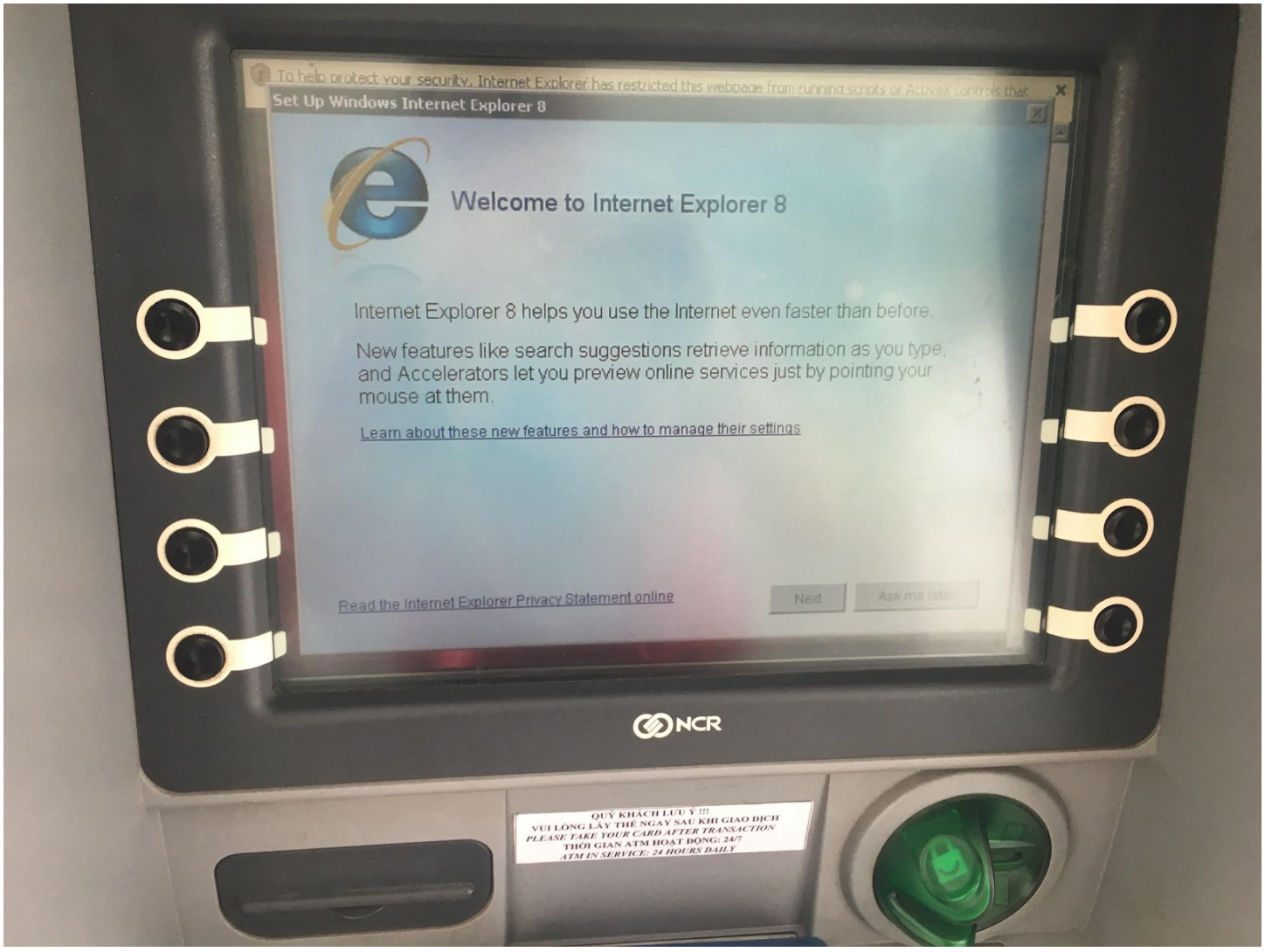
Security Now! #809 - 03-09-21

Hafnium

This week on Security Now!

This week we look into last week's critical Chrome update and also cover the wackiest-but-true Chrome extension of all time. We look at Google's new funding of Linux security development, a surprisingly un-dead long unheard from media player that just received a massive collection of updates, and yes, still another way of abusing Intel's latest processor microarchitecture. We need to update everyone on our Dependency Confusion topic from two weeks back because there's big news there, we have several bits of identical listener feedback all wanting to be sure that I knew something had happened. Then, we're going to cover the world's latest global crisis which we first mentioned as break news in the middle of last week's podcast. It was breaking then. It's badly broken now.

Not exactly confidence inspiring...



Browser News

As of last Tuesday, March 2nd, we're at Chrome 89.0.4389.72. This release included 47 security fixes and a bunch of bug hunters earned themselves some nice income. One researcher earned $10,000 for one fix and $7500 for another. And his 3rd is $TBD. Overall, there were a bunch of $10,000 awards, some $7500's, $5000's and $3000. As the main topic of today's podcast will highlight, the computer industry is experiencing a growing — not a shrinking — problem with attackable software. And industry companies have, by and large, responded by offering to pay for information regarding privately disclosed vulnerabilities discovered within their own software.

But it's not all happening in the light. As we know, there are also the likes of Zerodium who are offering big money for other people's vulnerabilities which they then turn around and resell to their shady clientele.

Among the 47 vulnerabilities fixed last Tuesday was a 0-day that was being exploited in the wild. Google was informed of the issue last month on February 11th by Alison Huffman of Microsoft Browser Vulnerability Research. Since Microsoft is now also on the Chromium bandwagon that makes sense. And Google included its fix in last week's Chrome update.

“Lay's Crispy Chips”

Before adding this one to today's show I had to do a double-take to make sure this was not an April Fool's posting. But it appears to be 100% legitimate. And I learned of this from BleepingComputer, who is very good about vetting their stories. It is a Chrome browser web extension called “Crispy Subtitles from Lay’s”:

https://chrome.google.com/webstore/detail/crispy-subtitles-from-lay/kokpckgbhcmeobdddflajfcpmmfhkekn

After this browser extension has been installed, anytime you are watching a YouTube video, and the system's AI-trained microphone detects the sound of crispy chips being eaten, YouTube captions will be automatically enabled to allow anyone — including yourself — to be able to obtain the video's dialog information over the sound of noisy chips being crunched. I’m not kidding.

Lawrence Abrams at BleepingComputer explains that to make it easier to watch YouTube videos, the creative agency Happiness Saigon partnered with Frito-Lay to create the 'Lay's Crispy Subtitles' browser extension that automatically enables YouTube captions when it detects you are eating chips. Lawrence says that to achieve this, Happiness Saigon trained an AI algorithm using 178 hours of recording people eating chips... from all over the world.

He said that BleepingComputer used the extension and was pleasantly surprised by how the extension immediately enabled YouTube captions when their microphone picked up the noisy sound of eating chips. He added that they had performed some tests with other food groups, including peanuts, carrots, and cereal. While peanuts and carrots were not noisy enough or crunchy enough, eating cereal also enabled captions in their tests to see what would trigger the extension. Lawrence concludes: "your results may vary depending on how noisy you eat your food."

Again, if this had been posted on April 1st it would be highly suspect. But Lawrence appears to be quite serious. And, of course, I checked-in with theGoogle Play Store and the extension exists.

Security News

Google funds Linux kernel security developers.
And speaking of Google doing good things for security, Google recently announced that they would be funding two Linux kernel developers' efforts as full-time maintainers exclusively focused on improving Linux security.

The Linux Foundation said: "While there are thousands of Linux kernel developers, all of whom take security into consideration as the due course of their work, this contribution from Google to underwrite two full-time Linux security maintainers signals the importance of security in the ongoing sustainability of open-source software."

Gustavo Silva and Nathan Chancellor, the two kernel developers funded through this initiative will exclusively focus on Linux kernel security development. Chancellor will triage and fix bugs in Clang/LLVM compilers. Silva will turn the elimination of several classes of buffer overflows into his full-time Linux development work.

The Linux Foundation noted that "Additionally, Gustavo Silva is actively focusing on fixing bugs before they hit the mainline, while also proactively developing defense mechanisms that cut off whole classes of vulnerabilities. He is consistently one of the top five most active kernel developers since 2017 and he has impacted 27 different stable trees, going all the way back to Linux v3.16."

The other funded developer, Nathan Chancellor tweeted: "I am very very thankful that I can get paid to do something that I love :) Thank you to everyone at Google and the Linux Foundation who was able to make this happen and I look forward to what this journey has in store for me. https://t.co/KIDjZsiCg7"

— Nathan Chancellor (@nathanchance) February 24, 2021

David Wheeler, the Linux Foundation's Director of Open Source Supply Chain Security said: "Ensuring the security of the Linux kernel is extremely important as it's a critical part of modern computing and infrastructure. It requires us all to assist in any way we can to ensure that it is sustainably secure. We extend a special thanks to Google for underwriting Gustavo and Nathan's Linux kernel security development work along with a thank you to all the maintainers, developers and organizations who have made the Linux kernel a collaborative global success."

WinAmp!!
WinAmp! What? WinAmp is still around? When was the last time anyone uttered the phrase "WinAmp" ?? I just couldn't pass up this blast from the past.

WACUP stands for the Winamp Community Update Project (WACUP) which recently released Preview version 1.0.20.7170 to provide an ungodly number of fixes and improvements to the venerable Winamp media player. Check out the changes and improvements page:

https://getwacup.com/preview/preview_1_0_20_7170.html

WACUP is a project by ex-Winamp developer Darren Owen aimed at fixing bugs and extending the Winamp v5.66 media player functionality through the program's plugin feature.

For thoSe who may not have been born when some of us were using WinAmp, one of the player's strongest features was that it includes a plugin system which allowed third-party developers to extend and modify the program's operation. The plugins range from new visualization tools, equalizers, and ways to change media's playback.

In any event, if anyone is interested, installing WACUP will also install WinAmp and get it all set to go on your modern operating system. WinAMp used to be commercial software. But after it was leaked online it was switched to Freeware. So, if you're curious about the way your elders once listened to music, you can grab it here: https://getwacup.com/

Lord of the Rings

https://arxiv.org/abs/2103.03443

We have yet another research paper describing a functional and, they claim, practical side-channel attack on the latest Intel processor microarchitecture. It is brilliant work. And it's worth noting, if only to place it on the record and into its proper context. The work is by a trio of intrepid researchers from the University of Illinois at Urbana-Champaign who will present their paper during the USENIX Security 2021 conference.

The paper is titled: "Lord of the Ring(s): Side Channel Attacks on the CPU On-Chip Ring Interconnect Are Practical" I'll start by sharing their paper's short Abstract:

> We introduce the first microarchitectural side channel at-tacks that leverage contention on the CPU ring interconnect.There are two challenges that make it uniquely difficult to exploit this channel. First, little is known about the ring interconnect's functioning and architecture. Second, information that can be learned by an attacker through ring contention is noisy by nature and has coarse spatial granularity. To address the first challenge, we perform a thorough reverse engineering of the sophisticated protocols that handle communication on the ring interconnect. With this knowledge, we build a cross-core covert channel over the ring interconnect with a capacity of over 4 Mbps from a single thread, the largest to date for across-core channel not relying on shared memory. To address the second challenge, we leverage the fine-grained temporal patterns of ring contention to infer a victim program's secrets.We demonstrate our attack by extracting key bits from vulner-able EdDSA and RSA implementations, as well as inferring the precise timing of keystrokes typed by a victim user.

Okay, so what is all of this "ring" business they're referring to? It's not something that we've ever talked about before.

There is so much more going on, on today's microprocessors that we are typically aware. Research into side-channel information leakage due to the fundamental architecture of modern processor design has been a big deal for the past years with a whole host of various attacks having — at least theoretically — been shown to create exploitable vulnerabilities. The result has been a bunch of Intel and AMD firmware updates.

This new attack leverages the bandwidth limitations and thus access contention among the chip's processor that can be made to vy for access to the so-called interconnect ring. The system on a chip (SoC) Ring interconnect is an on-die bus arranged in a ring topology which enables inter-component communication among the chips various subsystems. In this context they are known as “agents” and they include the chip's processor cores, the last level cache (LLC), any graphics cores, and other chip-level subsystems. Each of these ring agents communicates with the ring through what's called a ring stop.

So the researchers reverse-engineered Intel's proprietary and deliberately secret ring interconnect protocols to reveal the conditions under which two or more of these agents might force a ring contention. And that, in turn, allowed them to create a covert channel having a leakage capacity of 4.18 Mbps. The researchers noted that this is the largest to date covert channel rate for cross-core channels not relying upon shared memory.

One of the researchers said: “Unlike prior attacks, our attacks do not rely on sharing memory, cache sets, core-private resources or any specific uncore structures. As a consequence, they are hard to mitigate using existing 'domain isolation' techniques.”

To implement an attack, an adversary would measure the delay in memory access associated with a malicious process due to a saturation of bandwidth capacity caused by a victim process' memory accesses. The repeated latency in memory loads from LLC due to ring contention allows an attacker to use the measurements as a side-channel to leak key bits from vulnerable EdDSA and RSA implementations as well as reconstruct passwords by extracting the precise timing of keystrokes typed by a victim user.

Intel was not unduly upset or impressed by this. They built the ring and designed-in the handling for contention. So they fully realized that this could be done. And the weaponization of contention-based on-chip ring latencies is not something that is keeping Intel's microarchitecture engineers up at night. Nor should it keep us up at night.

As usual, the threat only presents in environments where malicious processes are sharing a processor with naïve processes containing secrets. And if any of our personal workstations have a malicious process loose then we already have bigger problems than some theoretical, difficult to leverage, cross-core leakage.

Dependency Contusion!

Our podcast two weeks ago was titled “Dependency Confusion.” And after explaining what that was all about, and noting that this is currently a fundamental design weakness and readily exploitable flaw within the industry's package managers and project build logic used pervasively throughout our industry, it was clear that this was going to be a large problem for quite some time. And that unfortunate expectation is being realized.

Stated as briefly as possible, Dependency Confusion amounts to a “by-design” backdoor into the internal build servers and private code repositories of a vast number of software publishers, both large and small.

The UK security firm, Sonatype, first blogged about the findings from their security

instrumentation last Monday. Their post was titled: "Newly Identified Dependency Confusion Packages Target Amazon, Zillow, and Slack; Go Beyond Just Bug Bounties"

https://blog.sonatype.com/malicious-dependency-confusion-copycats-exfiltrate-bash-history-and-etc-shadow-files

They write:

> Sonatype has identified new "dependency confusion" packages published to the npm ecosystem that are malicious in nature. These squatted packages are named after repositories, namespaces or components used by popular companies such as Amazon, Zillow, Lyft, and Slack.
>
> As previously reported by Sonatype, Alex Birsan's dependency confusion research disclosure led to copycat researchers publishing 275+ identical packages to the npm repo within 48 hours, in hopes of scoring bug bounties. The number then jumped to over 550 within the next few days.
>
> As of today, Sonatype's automated malware detection systems, part of Nexus Intelligence, has since identified well over 700 npm copycat packages based on Birsan's proof-of-concept packages.
>
> Although ethical hacking for bug bounties and spreading security awareness has its place and is even welcomed by the community as it keeps us all more secure, the copycat packages recently identified by Sonatype unfortunately crosses the line of what is deemed ethical.

Then, two days later, Sonatype's follow-up blog posting, last Wednesday, was titled: "PyPI and npm Flooded with over 5,000 Dependency Confusion Copycats"

https://blog.sonatype.com/pypi-and-npm-flooded-with-over-5000-dependency-confusion-copycats

This has predictably morphed, almost overnight, into an industry wide siege on the software industry's use of 3rd-party packaged libraries. And Sonatype notes that these are not all just white hats hoping to score a bounty. They wrote:

> ... as soon as these packages are installed automatically, because they share a name with an internal dependency (thereby exploiting "dependency confusion"), they exfiltrate the user's .bash_history file and /etc/shadow directory, and in some cases spawn a reverse shell. The /etc/shadow file is a successor to the /etc/passwd Linux file that maintains hashed password data of user accounts on a system.

So this exfiltration goes well beyond bug bounties. The original discoverer and publisher of this disaster, Alex Birsan, whose work we described two weeks ago, was extremely careful not to do anything that could be construed as malicious. He used simple DNS queries to indicate success. The fact that it's extremely difficult **NOT** to use this for malicious purposes demonstrates just how easy it is TO use it for malicious purposes.

Despite being warned about this by Alex's original work, in a follow-up experiment, Microsoft Teams was deeply penetrated. Last Thursday, Matt Austin, Director of Security Research at Contrast Security described his successful use of Dependency Confusion against Microsoft:

https://www.contrastsecurity.com/security-influencers/contrast-labs-reveals-dependency-confusion-vulnerability-in-microsoft-teams

This is enough of a problem that I think it's worth sharing a bit of Matt's description just to drive this point home. Part of what he wrote was:

> Dependency confusion involves the creation of packages on third-party libraries that have identical names to packages stored in a company's internal repository. The goal is to trick developers and automated systems into using the wrong library because of defaults configured into package managers that show a preference for external libraries over internal ones.
>
> While it should be difficult for external users—whether bad actors, legitimate developers, or security researchers—to even find the names of packages stored in internal repositories, Birsan was able to find the names of these packages in the applications themselves—and was able to replicate that accomplishment over at least 35 applications.
>
> Since Birsan's research was published, multiple reports of bad actors using the technique have started to appear. In addition, other security researchers, including myself, began exploring various Software-as-a-Service (SaaS)-based applications to see if we could find dependency confusion vulnerabilities that could be exploited by attackers.
>
> A member of the Contrast Labs team, I specialize in security research on applications created with the Electron framework, including Microsoft Teams. Previously, I have discovered and reported several remote code execution vulnerabilities in that application. Given my background, I decided to contribute to the effort by examining Teams to see if I could identify a dependency confusion vulnerability.
>
> I began by looking at the dependencies used by the Teams desktop application. In one section of the application, I saw a node.js package called "Optional Dependencies." Typically, the intent of optional dependencies is to provide a place to store binary files that are a part of the testing suite, but not the production application.
>
> The interesting thing about optional dependencies is that if they do not exist in the repository from which a developer is trying to pull them, the build will fail silently. Thus, developers must write code that is relatively defensive when optional dependencies exist. Specifically, one cannot "depend" on a dependency that is optional.
>
> Inside the Teams desktop application is a package file that lists the dependencies that the application needed in order to be built. When one downloads and installs the application, it does not reach out to the dependency managers. This is because every part of the application is built on a Microsoft server somewhere, after which it is shipped to customers.

Now back to what I discovered. When analyzing the package, I noticed a number of modules in both public and private repositories. I also saw that some of the private packages were not taking advantage of the scoping capability provided by npm. I noted that some of the optional dependencies did not exist on disk inside the package itself. This means that during the build process those dependencies would not be found in any repository—public or private. By design, these optional dependencies failed silently whenever the application attempted to pull them from both the public and private repositories.

I then selected one of the modules that was listed as an "optional dependency" and registered it on npm, setting the version to 0.0.1. I added a simple line of code in the install script of the package to alert me when it was installed. Once I was ready to deploy, I set the version of the module to match the version number from the application. At this point, I had two choices: bump the version number up by one and wait for Microsoft to update and pull in the new version, or keep it at the current version and hope that a build server would pull in a fresh copy of it (not using packagelock.json).

There are risks either way, but I decided to do what would be least impactful—keeping the version number the same. As soon as I set one of the dependencies to a high enough version, I started getting requests from the module being installed from a number of internal Microsoft IP addresses. The name of the module was relatively generic, but it was Microsoft-specific and not an overly common word.

I happened to be doing this research late at night. Based on that and other factors, I deduced that the responses were most likely from internal, automated resources within Microsoft that were pulling the dependencies and installing them—perhaps an automated continuous integration/continuous deployment (CI/CD) server. But it could also have been a live developer working a late shift in the U.S. or a daytime shift somewhere overseas.

Whether the responses I saw were automated or manual, the fact that I was able to generate this reaction poses significant risk. By taking advantage of the post-install script, I was able to

execute code in whatever environment this was being installed on. If attackers were to execute code the way I did on a build server for a desktop application update that was about to be distributed, they could insert anything they wanted into that update, and that code would go out to every desktop using Teams—more than 115 million machines. Such an attack could have monumental repercussions, potentially affecting as many organizations as the massive attack on the SolarWinds software factory that was revealed in December.

I wanted to drive home the point that **this will not be easy to fix** and that it is going to be a source of immense long-running pain for the entire industry. This would be similar to the first time someone observed that a deliberate buffer overflow could be abused to execute attacker-supplied code on the stack. Everyone knows it now. But there was a time when that had never occurred to anyone. And just look at what an enduring problem that has historically been. This is no less significant. A lazy and thoughtless approach to package dependency management has pervaded the software industry — it is already everywhere — and it's going to take a lot of work to pull it all out by the roots.

Listener Feedback

Aaron Wood / @The_Real_Abear
@SGgrc Microsoft Edge has native vertical tabs now, and it's awesome

Chris Ryan / @4given_p8ntblr
I don't know if you saw this @SGgrc, but MS Edge has vertical tabs native.

Robert Osorio / @RobertOsorio
@SGgrc Side tabs (ala TreeStyle) in Edge coming this month: engadget.com…

And yes, indeed, my Edge FINALLY has the little widget at the upper corner that instantly toggles between vertical and horizontal tabs. And it does remember the last setting it was in. So once properly set to the way man clearly intended all tabs to be arranged, it happily remains correct from then on. So ... a shout out thanks to all of our listeners who made sure I knew.

Hafnium

Exactly one week ago today, during last week's podcast, we shared the breaking news of Microsoft's emergency release of updates to patch four separate pre-authentication remote code execution (aka pre-auth RCE) exploits affecting their widely popular Exchange Servers 2013, 2016 and 2019. This has since exploded into another global crisis of truly epic proportion.

Today, we know that as many as hundreds of thousands of individual Exchange Servers have been under quiet attack since the beginning of January, so for two months before any alarm was raised. To give a sense for the impact of this, this past Saturday, Bloomberg News posted their coverage of this, beginning with the headline: "Microsoft Attack Blamed on China Morphs Into Global Crisis."

"Number of victims of Chinese attack continues to grow rapidly"
"White House warns companies to take threat "very seriously""

Bloomberg's coverage begins: "A sophisticated attack on Microsoft Corp.'s widely used business email software is morphing into a global cybersecurity crisis, as hackers race to infect as many victims as possible before companies can secure their computer systems. The attack, which Microsoft has said started with a Chinese government-backed hacking group, has so far claimed at least 60,000 known victims globally, according to a former senior U.S. official with knowledge of the investigation. Many of them appear to be small or medium-sized businesses caught in a wide net the attackers cast as Microsoft worked to shut down the hack."

Later in their article, Bloomberg wrote: "The Chinese hacking group, which Microsoft calls Hafnium, appears to have been breaking into private and government computer networks through the company's popular Exchange email software for a number of months, initially

targeting only a small number of victims, according to Steven Adair, head of the northern Virginia-based Volexity. The cybersecurity company helped Microsoft identify the flaws being used by the hackers for which the software giant issued a fix on Tuesday."

The truth is, Volexity was the discoverer of this whole mess, so we'll switch over to Volexity in a moment. But Bloomberg quoted Charles Carmakal, a senior VP at FireEye as saying: "The good guys are getting tired"; which I thought was such a sad statement of today's state of affairs. We're still dealing with the aftermath of the SolarWinds breach, and now this.

And the attack rate exactly followed the discovery & patch model we've often talked about here. Bloomberg wrote: "Initially, the Chinese hackers appeared to be targeting high value intelligence targets in the U.S. Then, about a week ago, everything changed. Other unidentified hacking groups began hitting thousands of victims over a short period, inserting hidden software to give them access later."

Volexity's Steve Adair said: "They went to town and started doing mass exploitation -- indiscriminate attacks compromising exchange servers, literally around the world, with no regard to purpose or size or industry. They were hitting any and every server that they could."

This is the attacker rationale that we've noted in the past. So long as exploits remain unknown, the attacker's optimal strategy is quiet steathful surgical strikes. Since discovery will happen sooner or later, there will always be time to switch into mass exploitation mode. So, setup for mass exploitation so that you're ready when that time comes. But until then, carefully target the highest value victims.

Stealth mode had been underway for two months. And lord only knows how many high value victims were compromised during that time. So that damage had been done. Then, the moment Microsoft went public with patches the jig was up and the optimal strategy then became to exploit as many systems as possible -- including implanting persistent backdoors -- before those systems can have the newly available patches applied.

Last Saturday the 6th, Kevin Beaumont, who is now with Microsoft tweeted from his "@GossiTheDog" account: *"A single server in the MailPot honeypot has been exploited with these vulnerabilities 5 times today, as a data point. So far nobody has actually done anything with those webshells, just planted them."*

The primary takeaway is that patching after the announcement was almost certainly closing the barn door after the horses had left. The mass automated exploitation that exploded early last week was a race to install webshell backdoors into all available vulnerable servers before those patches could be applied. So any admins who think "Whew! Glad we're okay now! Everything looks fine!" are probably fooling themselves. Everything will look fine, but might not be.

We were just talking about webshells: About how they leverage a website's scripting engines against themselves. How trivial they are to design and implant, yet how diabolical they can be once they are in place. They turn any web server into a slave by equipping it with a new set of command-following capabilities of the attacker's design. And they hide among any web site's gazillion other mystery files. And that's exactly what the mass exploitation effort was rapidly implanting into every available exchange server.

What does one of the early discoverers of this latest nightmare explain?

Steven Adair, Volexity President said: *"We've worked on dozens of cases so far where web shells were put on the victim system back on Feb. 28 [before Microsoft announced its patches], all the way up to today. Even if you patched the same day Microsoft published its patches, there's still a high chance there is a web shell on your server. If you're running Exchange and you haven't patched this yet, there's a very high chance that your organization is already compromised."*

I just identified Volexity's and Steven Adair as "one of" the early discoverers. And here, amid what is now being called a global crisis, we learn that, believe it or not, Microsoft was credibly informed of these threats by three independent researchers as far back as early January. With the second and third reports being of detected use of these vulnerabilities in the wild. Yet Microsoft apparently felt no great urgency to act on the world's behalf.

On January 5th, a researcher tweeting as Orange Tsai posted:

The tweet reads: "Just report a pre-auth RCE chain to the vendor. This might be the most serious RCE I have ever reported! Hope there is no bug collision or duplicate." I assume he means that he's hoping to collect a big bug bounty on this one.

After everything went public, and the you-know-what has hit the fan. Orange Tsai, who is a member of DEVCORE, has taken their "ProxyLogon" (https://proxylogon.com/) disclosure site public. And from them we learn of the timeline of their interaction with Microsoft:

October 01, 2020	DEVCORE started reviewing the security on Microsoft Exchange Server
December 10, 2020	DEVCORE discovered the first pre-auth proxy bug (CVE-2021-26855)
December 27, 2020	DEVCORE escalated the first bug to an authentication bypass to become admin
December 30, 2020	DEVCORE discovered the second post-auth arbitrary-file-write bug (CVE-2021-27065)
December 31, 2020	DEVCORE chained all bugs together to a workable pre-auth RCE exploit
January 05, 2021	DEVCORE sent (18:41 GMT+8) the advisory and exploit to Microsoft through the MSRC portal directly
January 06, 2021	MSRC acknowledged the pre-auth proxy bug (MSRC case 62899)
January 06, 2021	MSRC acknowledged the post-auth arbitrary-file-write bug (MSRC case 63835)
January 08, 2021	MSRC confirmed the reported behavior
January 11, 2021	DEVCORE attached a 120-days public disclosure deadline to MSRC and checked for bug collision
January 12, 2021	MSRC flagged the intended deadline and confirmed no collision at that time
February 02, 2021	DEVCORE checked for the update
February 02, 2021	MSRC replied "they are splitting up different aspects for review individually and got at least one fix which should meet our deadline"
February 12, 2021	MSRC asked the title for acknowledgements and whether we will publish a blog
February 13, 2021	DEVCORE confirmed to publish a blog and said will postpone the technique details for two weeks, and will publish an easy-to-understand advisory (without technique details) instead
February 18, 2021	DEVCORE provided the advisory draft to MSRC and asked for the patch date
February 18, 2021	MSRC pointed out a minor typo in our draft and confirmed the patch date is 3/9
February 27, 2021	MSRC said they are almost set for release and wanted to ask if we're fine with being mentioned in their advisory
February 28, 2021	DEVCORE agreed to be mentioned in their advisory
March 03, 2021	MSRC said they are likely going to be pushing out their blog earlier than expected and won't have time to do an overview of the blog
March 03, 2021	MSRC published the patch and advisory and acknowledged DEVCORE officially
March 03, 2021	DEVCORE has launched an initial investigation after informed of active exploitation advisory from Volexity
March 04, 2021	DEVCORE has confirmed the in-the-wild exploit was the same one reported to MSRC
March 05, 2021	DEVCORE hasn't found concern in the investigation so far

I mentioned three independent researchers. DEVCORE began their analysis of Exchange Server on October 1st and found the first of several remotely exploitable problems on December 10th. And by the end of December they had designed a chained exploit capable of pre-auth RCE. But, independently, Reston, Virginia based Volexity, discovered attacks using these flaws on Jan. 6. They watched and characterized it, and officially informed Microsoft about it on Feb. 2. And the third research group, the Danish security firm Dubex, says it first saw their clients being hit on Jan. 18, and they reported their incident response findings to Microsoft on Jan. 27.

So, Microsoft first learned of this as a CRITICAL, five-alarm, easy-to-exploit pre-authentication attack against Exchange Server at the start of January. Then, by the end of January (the 27th) and the start of February (the 2nd) they learned from two additional independent firms that the vulnerabilities were now under active exploitation. But they did not act. The only way I can explain it is that perhaps they were hoping to bury these CRITICAL 0-day vulnerability updates in March's Patch Tuesday, which is today... and hope that no one would notice. But if so, why didn't they?

And recall which servers were patched? Brian Krebs', in his post mortem construction of a timeline made an interesting and chilling observation. Brian wrote: "On Mar. 2, Microsoft patched four flaws in Exchange Server 2013 through 2019. Exchange Server **2010** is no longer supported, but the software giant made a "defense in depth" exception and gave Server 2010 users a freebie patch, too. That means the vulnerabilities the attackers exploited have been in the Microsoft Exchange Server code base for more than ten years." Yikes.

Volexity has a fully detailed write-up with indicators of compromise, the IP addresses to which data was being exfiltrated, and so on. But they begin with a short narrative that I think our listeners will find illuminating...

https://www.volexity.com/blog/2021/03/02/active-exploitation-of-microsoft-exchange-zero-day-vulnerabilities/

> In January 2021, through its Network Security Monitoring service, Volexity detected anomalous activity from two of its customers' Microsoft Exchange servers. Volexity identified a large amount of data being sent to IP addresses it believed were not tied to legitimate users. A closer inspection of the IIS logs from the Exchange servers revealed rather alarming results. The logs showed inbound POST requests to valid files associated with images, JavaScript, cascading style sheets, and fonts used by Outlook Web Access (OWA). It was initially suspected the servers might be backdoored and that webshells were being executed through a malicious HTTP module or ISAPI filter. As a result, Volexity started its incident response efforts and acquired system memory (RAM) and other disk artifacts to initiate a forensics investigation. This investigation revealed that the servers were not backdoored and uncovered a zero-day exploit being used in the wild.
>
> Through its analysis of system memory, Volexity determined the attacker was exploiting a zero-day server-side request forgery (SSRF) vulnerability in Microsoft Exchange (CVE-2021-26855). The attacker was using the vulnerability to steal the full contents of several user mailboxes. This vulnerability is remotely exploitable and does not require authentication of any kind, nor does it require any special knowledge or access to a target environment. The attacker only needs to know the server running Exchange and the account from which they want to extract e-mail.

Additionally, Volexity is providing alternative mitigations that may be used by defenders to assist in securing their Microsoft Exchange instances. This vulnerability has been confirmed to exist within the latest version of Exchange 2016 on a fully patched Windows Server 2016 server. Volexity also confirmed the vulnerability exists in Exchange 2019 but has not tested against a fully patched version, although it believes they are vulnerable. It should also be noted that this vulnerability does not appear to impact Office 365.

Following the discovery of CVE-2021-26855, Volexity continued to monitor the threat actor and work with additional impacted organizations. During the course of multiple incident response efforts, Volexity identified that the attacker had managed to chain the SSRF vulnerability with another that allows remote code execution (RCE) on the targeted Exchange servers (CVE-2021-27065). In all cases of RCE, Volexity has observed the attacker writing webshells (ASPX files) to disk and conducting further operations to dump credentials, add user accounts, steal copies of the Active Directory database (NTDS.DIT), and move laterally to other systems and environments.

It is difficult **not** to hold Microsoft responsible for what appears to be a sluggish response to a VERY clear and present massively widespread CRITICAL danger to all users of their Exchange Server product. Yeah, Microsoft's big. But we are often seeing other big companies, like Apple and Google, responding in days and even sometimes in hours. This took months. With the foreseeable result that a phenomenal number of organizations are now likely compromised — with no idea that they have been. Even if they've patched, their Exchange web servers are now likely to be hosting a dormant backdoor webshell.

BleepingComputer also has coverage, as of last Sunday, of Microsoft's just-updated-for this-Nightmare standalone MSERT server scanner which has been updated to detect any of seven webshells that might now be present on a compromised exchange server. It can be downloaded for either 32-bit or 64-bit machines. The link to its homepage is in the show notes and it's also this week's GRC.SC shortcut. So it's https://grc.sc/809

https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/safety-scanner-download

Note that Microsoft's safety scanner will only run for 10 days after download to assure that older releases are not trusted. So you can always grab an update either by using this week's 809 shortcut, or by bookmarking the redirected page after once using this week's shortcut.

And as if all of this wasn't bad enough. I have a **VERY** important note about the likely failure of manually installed Exchange Server patches: *If the updates are NOT run as an administrator, believe it or not, the patching will fail silently, providing no warning feedback and leaving the system unpatched and unprotected!*

Microsoft said that the installer will not display any errors or messages to let you know that the Exchange security updates have not been installed correctly. They wrote: "This issue occurs on servers that are using User Account Control (UAC). The issue occurs because the security update doesn't correctly stop certain Exchange-related services." To work around this known issue, Microsoft recommends installing the security updates as an administrator. So... if anyone here might not have done that last week, you'll want to double check!

I have a link down at the end of the show notes to Microsoft's page titled: "Repair failed installations of Exchange Cumulative and Security updates" which was last updated yesterday.

https://docs.microsoft.com/en-us/exchange/troubleshoot/client-connectivity/exchange-security-update-issues

They write: "This article describes the methods to verify the installation of Microsoft Exchange Server Cumulative Updates (CUs) and Security Updates (SUs) on your servers, lists known issues that you might encounter when installing CUs and SUs, and provides resolutions to fix the issues."

That page provides links to scripts that can be used to check for Indicators of Compromise (IOCs) and also instructions for verifying the installation of CUs & SUs.

Use the following script which automates testing for all four vulnerabilities described in the Microsoft Threat Intelligence Center (MSTIC) blog post. The script provides a progress bar and performance tweaks to make the test for CVE-2021-26855 test run fast. Download the latest version of the script from the Exchange Support GitHub repository https://github.com/microsoft/CSS-Exchange/tree/main/Security

Next Week on Security Now!

Next week: Inside Google's plan to replace 3rd party tracking cookies with "FLoC" "Federated Learning of Cohorts" (what an awful name). FLoC is quite controversial, and next week we'll learn all about it.