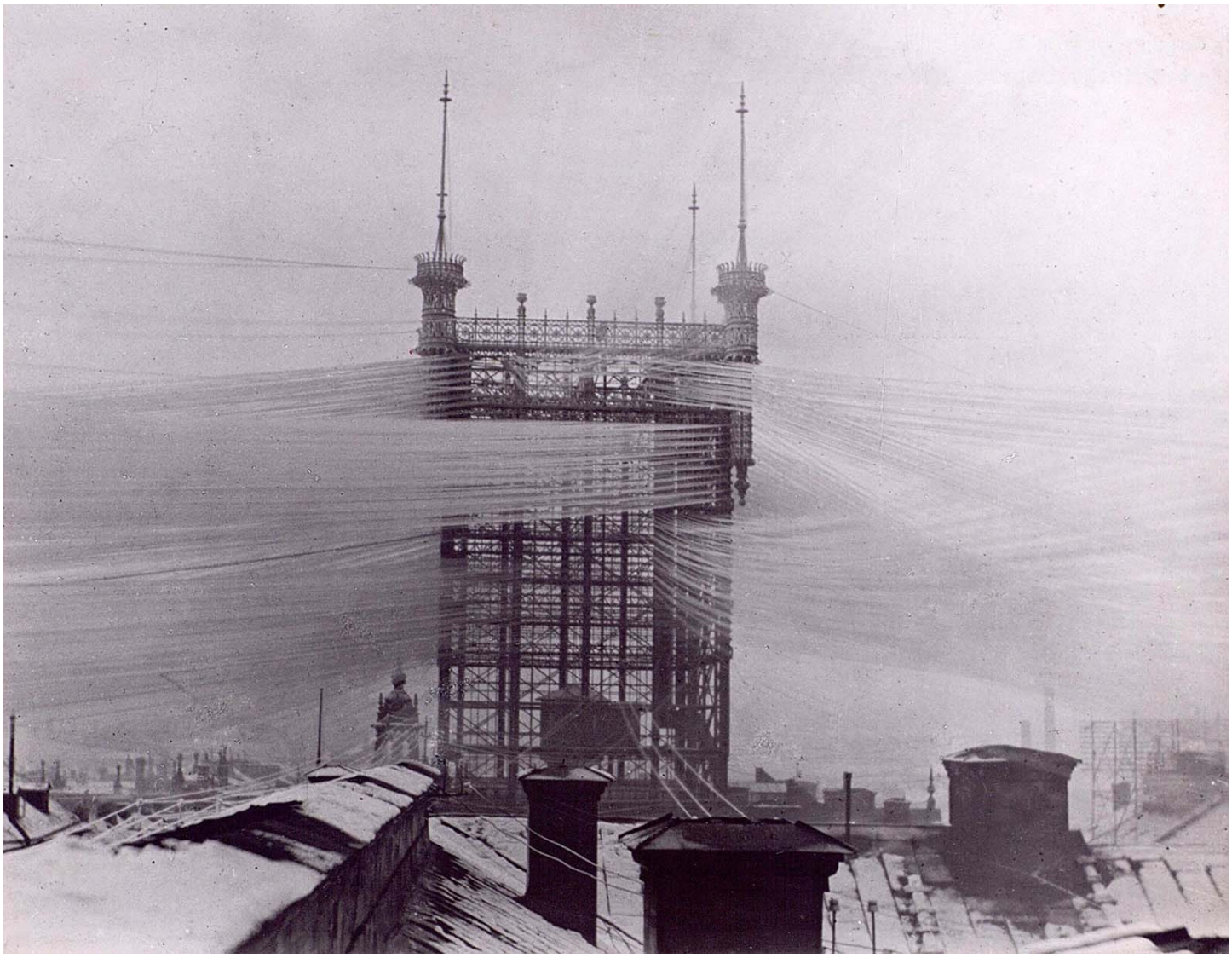
Security Now! #833 - 08-24-21

Microsoft’s Reasoned Neglect

This week on Security Now!

This week we briefly look at Firefox's plan to block unsecured downloads. We examine the threat posed by T-Mobile's massive and deep data breach and what current and past customers of T-Mobile should do. We look at three additional so-called “Overlay Networks” in addition to TailScale, and also at the consequences of another Orange Tsai Microsoft Exchange Server exploit chain discovery. We’ll also examine a simple-to-make flaw in the Razer gaming mouse installer, cover another worrisome IoT protocol screw-up, and share a couple of feedback notes and a question from our listeners. Then I want to conclude by following up on last week's discussion of Microsoft's apparent culpable negligence with a proposed explanation of their behavior and motivation which fits the facts so well that it becomes Reasoned Neglect.



The old “Telefontornet” telephone tower in Stockholm, Sweden, with approximately 5,500 telephone lines c. 1890.
https://rarehistoricalphotos.com/the-stockholm-telephone-tower-1890/

In the late 19th century, the miracle device called the telephone had been invented but the simple concept of undergrounding telephone cables had eluded engineers. Due to technical limitations of the earliest phone lines, every telephone required its own physical line strung between a house or business to a phone exchange where the call was manually connected by a live operator. The somewhat quixotic result of so many individual lines was the construction of elaborate and unsightly towers that carried hundreds to thousands of phone lines through the air.

In Stockholm, Sweden, the central telephone exchange was the Telefontornet, a giant tower designed around 1890 that connected some 5,000 lines which sprawled in every direction across the city. Just by looking at historical photos it's easy to recognize the absurdity and danger of the whole endeavor, especially during the winter months. Everything that could possibly go wrong did. From high winds to ice storms and fires, the network was extremely vulnerable to the elements. Luckily, phone networks evolved so rapidly that by 1913 the Telefontornet was completely decommissioned in favor of much simpler technology. The remaining shell stood as a landmark until it too caught fire in 1953 and was torn down.

Telephone service was expensive at that time, and only the wealthy could afford it. In Sweden, the first public telephone exchange was opened by the Bell Telephone Company. It originally had only 121 subscribers. The telephone company charged subscribers between 160 and 280 Swedish Krona, depending on the subscriber's location and distance to the exchange. This was equivalent to paying a subscription fee of 9,000 to 16,000 Krona (USD 1,100 to USD 1,966) in today's value, which was a very high rate.

Browser News

Firefox soon to be blocking mixed-content downloads by default

Back in 2020, as we discussed at the time, Chrome and the Chromium browsers gradually made the switch to blocking mixed-content downloads as they moved from release 81 to 88. This meant that the browser would step in and block a user who clicked on a download link for a file which would, if it weren't blocked, be delivered over unauthenticated and unencrypted HTTP when the page referring to that download was, itself, delivered over HTTPS.

So, a feature which is already present but is not yet enabled by default will soon be flipped on for everyone. Once that's done...

- All HTTP files download from an HTTPS page will be blocked with a message in the Firefox Download Center (CTRL+J).
- An option will be available to let users allow the download if they choose to.
- HTTP file downloads from HTTP pages will not be blocked.
- Directly accessed HTTP download links (copy-pasted in the Firefox address bar) will not be blocked.

The feature is currently present and enabled in the Firefox Beta, Developer, and Nightly editions. Based on current Firefox bug tracker entries, the feature is expected to be activated for all Firefox users in v92, scheduled for a formal release at the start of September 2021.

This can be turned on now for Firefox by going to "about:config" and place "insecure" into the search box. The first item to be shown is "dom.block_download_insecure" which you'll want to switch to "true".

Security News

The news from T-Mobile is all bad.

Last week T-Mobile confirmed their latest data breach, making it the fifth data breach in four years. There were two previous attacks in 2020, one in 2019 and the first in 2018. But this most recent breach is the largest by far... and the numbers of affected customers keeps growing. The most recent update reveals that the cyberattack exposed over 54 million individuals' data.

Last weekend, a threat actor began selling the personal information of 100 million T-Mobile customers on a hacking forum for six bitcoin (~$280K). The hacker claimed that the stolen database contains the data for approximately 100 million T-Mobile customers. The exposed data can include customers' IMSI, IMEI, phone numbers, customer names, security PINs, Social Security numbers, driver's license numbers, and date of birth. In other words, the keys to the identity theft kingdom. The database was said to have been stolen approximately three weeks ago and contains customer data dating as far back as 2004. In an interview with the hacker, BleepingComputer's founder Lawrence Abrams reported that the hacker said: “Their entire IMEI history database going back to 2004 was stolen.”

After the data first went up for sale, T-Mobile later confirmed that some of its servers had been hacked and began investigating what customer data was exposed. Last Tuesday, on August 17th, T-Mobile first said that the personal information of 48.6 million individuals was exposed during the attack. They later updated that to include an additional 6 million customers or prospective customers who were also affected by the attack. And T-Mobile also confirmed that the attackers also stole their customers' IMSI and IMEI numbers.

- 13.1 million current T-Mobile postpaid customer accounts that included first and last names, date of birth, SSN, and driver’s license/ID information.
- 40 million former or prospective T-Mobile customers, including first and last names, date of birth, SSN, and driver’s license/ID information.
- 667,000 accounts of former T- Mobile customers exposing customer names, phone numbers, addresses and dates of birth compromised.
- 850,000 active T-Mobile prepaid customer names, phone numbers and account PINs were exposed.
- 52,000 names related to current Metro by T-Mobile accounts may have been included.

Identity theft is one of those things that can really screw up one's life. You need to prove that it wasn't you who applied for and received credit under your name — when the other person provided all of the personal information that only you are presumed to have. Such cretins immediately ran up massive charges under your name using, and destroying your credit. There are tons of horror stories about the mess this has caused for people. And what's needed to apply for credit in someone else's name? Exactly the data that has just been exposed for tens of millions of T-Mobile customers.

This is why the absolutely #1 best advice I have — the advice I give to anyone and everyone — is to simply run with permanent locks on your accounts at all three of the credit reporting bureaus: Experian, TransUnion, Equifax. I've had all three locked for me since I first talked about this years ago. This is easily done for someone who is not routinely applying for credit. And it's also recently become easier for those who do need to occasionally unfreeze their credit worthiness.

A couple of years ago I realized that I was losing money by not using an Amazon brand credit card for my many purchases through Amazon since they were providing a couple of points of discount for purchases made through their own card. Why would I not take advantage of that? So I had the occasion to need to lift the locks. What I discovered was that all three of the agencies now offer a convenient 10-day unlock with automatic relock. So I told them all to start the 10-day counter. I applied for the Amazon card, qualified through whichever one of the three Amazon uses, then all of them were relocked.

With accounts locked in this way, no would-be creditor is able to query for my credit and thus allow a thief to open a credit account in my name. If you have not done it. If you are not continually needing to have creditors accessing your credit with the big three, then why not lock those agencies down? It's trivial to do and it buys a lot of peace of mind.

Introducing ProxyLogon's kissing cousin, ProxyShell

The Taiwanese security researcher known as Orange Tsai (whose work we've covered before) unveiled his “ProxyShell” exploit chain during the Pwn2Own 2021 hacking contest a few months ago in April of this year. The chain invokes three unique and original exploits, and they are doozies (that's the technical term):

- CVE-2021-34473 provides a mechanism for pre-authentication remote code execution, enabling malicious actors to remotely execute code on an affected system.
- CVE-2021-34523 enables malicious actors to execute arbitrary code post-authentication on Microsoft Exchange servers due to a flaw in the PowerShell service not properly validating access tokens.
- CVE-2021-31207 enables post-authentication malicious actors to execute arbitrary code in the context of SYSTEM and write arbitrary files.

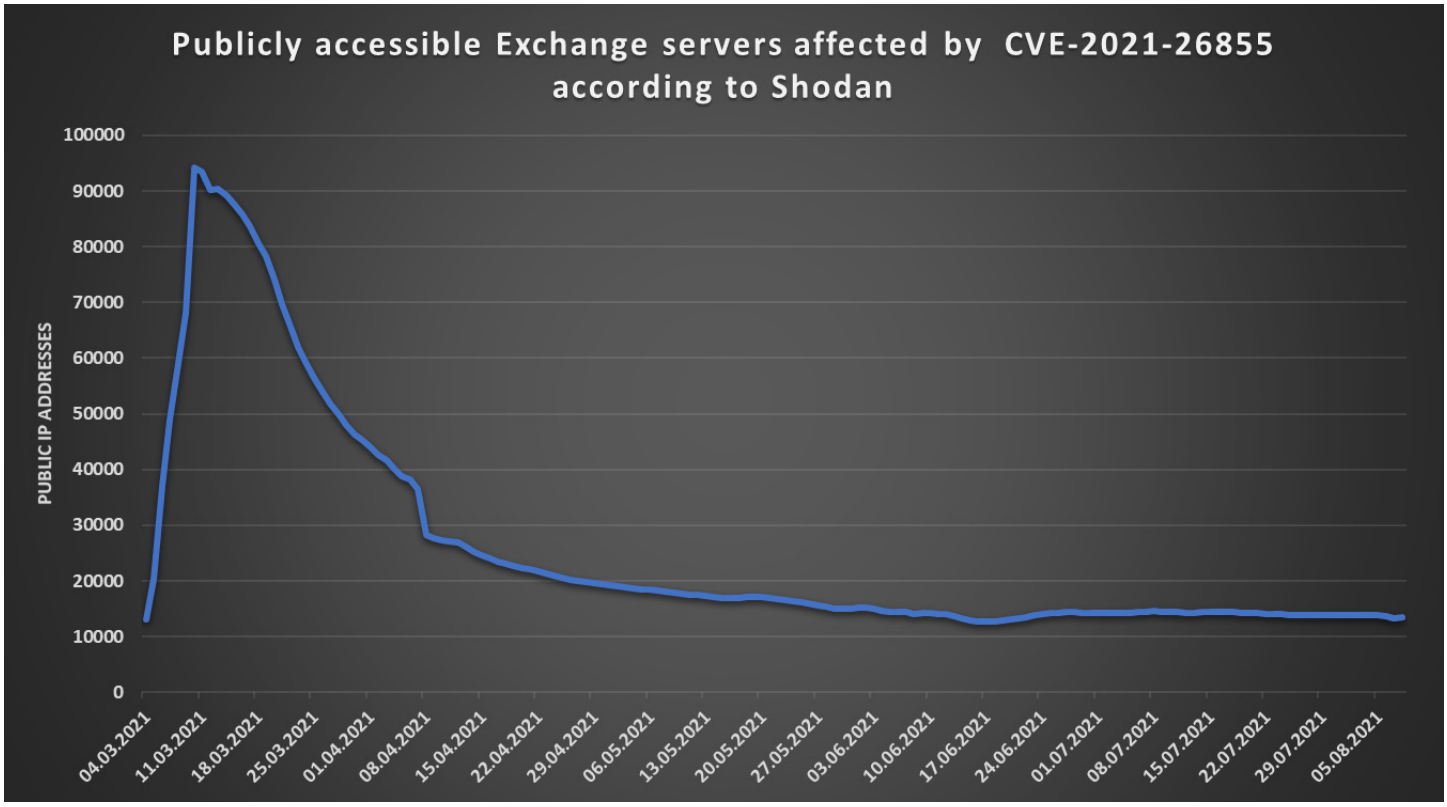
So, we have a pre-auth remote code execution exploit and a pair of post-authentication exploits one of which enables an attacker to execute arbitrary code as ROOT and to write arbitrary files. Do you imagine that a talented hacker might make something of those?

Orange Tsai used this trio to completely take over a Microsoft Exchange Server remotely, and in so doing earned himself $200,000 for a successful server compromise. Of course, part of the deal of Pwn2Own is that the details of any vulnerabilities used are turned over to the publishers of the impacted software. So, in this case the details of the exploit chain were immediately shared with Microsoft, following which Microsoft promptly patched the three vulnerabilities in May and July this year.

Then during his follow-up talk at BlackHat during the first week of August, Orange Tsai disclosed additional information about the attack. That information was leveraged and shortly followed by

the publication of additional technical information about it by other researchers who managed to recreate it. After that, it didn't take the bad guys long to start looking for vulnerable machines.

And , sure enough, as with the earlier ProxyLogon and ProxyOracle disclosures in March and April, not all server administrators jumped on those patches. Consequently, a scan performed two Sundays ago, on August 8 by SANS Security, two days after the publication of ProxyShell's proof-of-concept code, discovered more than 30,400 Exchange servers from a total of 100,000 systems had yet to be patched and remained vulnerable to attacks.



https://isc.sans.edu/forums/diary/ProxyShell+how+many+Exchange+servers+are+affected+and+where+are+they/27732/

In other words, Microsoft patches this in May and July. And one month after July's patch Tuesday, a month after any Exchange Servers receiving updates would have received them, 30,400 servers remain vulnerable.

The initial pre-exploitation started with scans for vulnerable systems, which then turned into actual attacks over the past weekend, according to honeypot logs collected by security researchers Rich Warren and Kevin Beaumont. After that attacks intensified last week, and a new ransomware operation known as LockFile began using the ProxyShell exploit as a way to enter corporate networks.

Kevin Beaumont tweeting from his @GossiTheDog account wrote last Friday: *"ProxyShell is now being used to drop corporate ransomware (as is PetitPotam), same IP and actor as in this thread. Myself and @buffaloverflow have been watching them."*

On Friday, security firm Huntress Labs said it scanned Microsoft Exchange servers that have been hacked using ProxyShell and found more than 140 different web shells on more than 1,900

Exchange servers. Recall that a web shell is the most trivial-to-plant and difficult-to-spot bit of code. On any server using server-side script interpretation—and Microsoft's Active Server Pages (.aspx) is ubiquitous on Windows servers and effectively impossible to get rid of—just a short and simple page of text can be written to implement a listening web shell. This fact really demonstrates the security short-sidedness that's inherent in server side scripting. It's another classic example where ease of use has vastly outstripped the teachings of security principles.

Kyle Hanslovan, the CEO and co-founder of Huntress Labs said: "Impacted organizations thus far include building manufacturers, seafood processors, industrial machinery, auto repair shops, a small residential airport, and more."

The long tail on that "servers remaining vulnerable" chart make clear that it's going to be a long time—if ever—until these previously installed instances of Exchange Server stop being infested with malicious malware.

The Razer mouse hack

Probably the most noise was made during the past week, this weekend, over a clever hack that someone discovered involving the installation of drivers for the Razer mouse... though describing it as a 0-day vulnerability, as much of the tech press has, seems a bit much to me.

For those who don't know, "Razer" is a popular manufacturer of gaming-oriented peripherals — mice and keyboards. So, as Windows will, when a device is encountered for which it lacks currently installed drivers, it fetches them on-the-fly over the Internet from its vast repository of driver installation packages which have been written and submitted by the vendors of those 3rd party products. We all know this as "plug-and-play" or, less charitably as "plug and pray."

In any event, when a Razer device is first introduced to Windows, the OS automatically obtains and begins to install the Razer Synapse software onto that local machine. Razer's software allows their users to configure and customize their hardware, setting up macros, mapping buttons, and so on. And Razer's products are quite popular, being used by more than 100 million users worldwide.

So, as it happened, a security researcher using the handle "jonhat" discovered an exploitable escalation of privilege mistake that had been made in the driver installation flow which allows users whose local privilege may have been tightly locked down, to escape that lock and obtain the full local privileges of the root SYSTEM account. And to do so with somewhat surprising ease.

Since device installation needs to make deep changes to a system, Windows runs these installers with SYSTEM privileges so that they'll be able to make these changes. And by the way, this is another horrible kludge in Windows. I learned during the development of SQRL's self- installation system that Windows employs a messy heuristic here. It "sniffs" programs being run—I kid you not—looking for strings such as "setup" or "installer" and treats such programs differently. What a disaster.

Anyway... the program RazerInstaller.exe will be downloaded and executed by a Windows OS process running with SYSTEM privileges. This allows the RazerInstaller.exe child process to inherit the same privileges as its parent. That's necessary to enable the installer to do what it

might need to. Okay. Everything's fine so far. But the mistake the Razer developers made was in then causing the setup wizard to allow the user to specify the folder into which the software should be installed **without restricting its rights:**

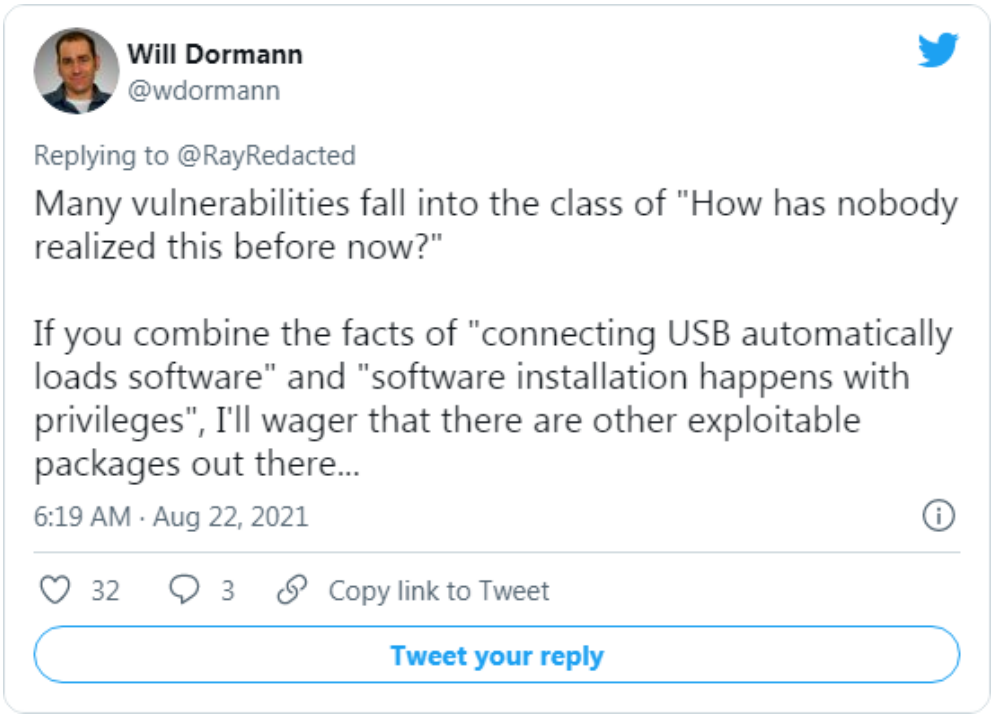
If the user elects to alter the default installation destination, Windows will present the "Choose a Folder" dialog. But in this instance, this dialog will have **also** inherited the installer's full SYSTEM privileges. So, if a clever user holds down Shift and right-clicks on that dialog, a context menu will pop up, and >>SURPRISE!<< among its various options is "Open PowerShell window here." Whoopsie!! If that option is selected, sure enough, a PowerShell window is opened having the same rights as the dialog that spawned **it**... namely, full SYSTEM privileges.

So, in short, the mistake of not reducing the rights of a dialog spawned by a privileged installer, which any locked-down user having no rights can cause to be launched just by connecting any Razer mouse, grants that user full and unrestricted ROOT level access on any Windows machine to which they have physical access.

Now, having discovered this important oversight, "jonhat" attempted to do the right thing with his discovery. He reached out and tried to responsibly contact and notify Razer. They blew him off. So, after not receiving any response from Razer, he posted his discovery to Twitter last Saturday. "jonhat" tweeted:

> *Need local admin and have physical access? - Plugin a Razer mouse (or the dongle) - Windows Update will download and execute RazerInstaller as SYSTEM - Abuse elevated Explorer to open Powershell with Shift+Right click. Tried contacting @Razer but no answers. So here's a freebie*

Now, if you're thinking: "Gee... that seems like an easy mistake for someone to make. I wonder whether any other installers do that too?" ... you would be in good company. The following day, last Sunday, CERT/CC's Vulnerability Analyst Will Dormann observed and tweeted:



IoT News

A critical ThroughTek SDK flaw enables IoT spying

So it turns out that somewhere on the order of 83 million actively in use IoT devices which are streaming live audio and video, are doing so over a little known and not very secure peer to peer network that goes by the name Kalay.

Last Tuesday, the guys at Fireeye from their Mandiant security team posted the news of their discovery under the title "Mandiant Discloses Critical Vulnerability Affecting Millions of IoT Devices". In their posting they explain:

Today, Mandiant disclosed a critical risk vulnerability in coordination with the Cybersecurity and Infrastructure Security Agency ("CISA") that affects millions of IoT devices that use the ThroughTek "Kalay" network. This vulnerability, discovered by researchers on Mandiant's Red Team in late 2020, would enable adversaries to remotely compromise victim IoT devices, resulting in the ability to listen to live audio, watch real time video data, and compromise device credentials for further attacks based on exposed device functionality. These further attacks could include actions that would allow an adversary to remotely control affected devices.

At the time of writing this blog post, ThroughTek advertises having more than 83 million active devices and over 1.1 billion monthly connections on their platform. ThroughTek's clients include IoT camera manufacturers, smart baby monitors, and Digital Video Recorder ("DVR") products. Unlike the vulnerability published by researchers from Nozomi Networks in May 2021 (also in coordination with CISA), this latest vulnerability allows attackers to communicate with devices remotely. As a result, further attacks could include actions that would allow an adversary to remotely control affected devices and could potentially lead to remote code execution.

[I know I sound like a broken record about this. But let me just say again: You do not want any of your IoT devices sharing a network with your PCs.]

The Kalay protocol is implemented as a Software Development Kit ("SDK") which is built into client software (e.g. a mobile or desktop application) and networked IoT devices, such as smart cameras. Due to how the Kalay protocol is integrated by original equipment manufacturers ("OEMs") and resellers before devices reach consumers, Mandiant is unable to determine a complete list of products and companies affected by the discovered vulnerability.

This vulnerability has been assigned a CVSS3.1 base score of 9.6 and is tracked as CVE-2021-28372 and FEYE-2021-0020. This blog post discusses the Kalay network and CVE-2021-28372 at a high level. It also includes recommendations from ThroughTek and Mandiant, along with mitigation options.

Mandiant would like to thank both CISA and ThroughTek for their coordination and support in releasing this advisory.

https://www.fireeye.com/blog/threat-research/2021/08/mandiant-discloses-critical-vulnerability-affecting-iot-devices.html

The way the researchers penetrated an unknown and opaque network of devices is kinda cool:

Mandiant researchers analyzed ThroughTek's Kalay protocol using two different approaches. First, the researchers selectively downloaded and disassembled applications from both the Google Play Store and Apple App Store that included ThroughTek libraries. These libraries typically did not contain debugging symbols, which required the team to also perform dynamic analysis with tools such as Frida, gdb, and Wireshark.

In addition, Mandiant purchased various Kalay-enabled devices. The team performed local and hardware-based attacks to obtain shell access, recover firmware images, and perform additional dynamic testing. These techniques included identifying UART/JTAG interfaces, performing chip-off attacks, and exploiting other debugging functionality present on the devices.

Over the course of several months, the researchers developed a fully functional implementation of ThroughTek's Kalay protocol, which enabled the team to perform key actions on the network, including device discovery, device registration, remote client connections, authentication, and most importantly, process audio and video ("AV") data. Equally as important as processing AV data, the Kalay protocol also implements remote procedure call ("RPC") functionality. [Ooof.... what could possibly go wrong?] *This varies from device to device but typically is used for device telemetry, firmware updates, and device control.*

Having written a flexible interface for creating and manipulating Kalay requests and responses, Mandiant researchers focused on identifying logic and flow vulnerabilities in the Kalay protocol. The vulnerability discussed in this post affects how Kalay-enabled devices access and join the Kalay network. The researchers determined that the device registration process requires only the device's 20-byte uniquely assigned identifier (called a "UID" here) to access the network. In Mandiant's testing, this UID was typically provided to a Kalay-enabled client (such as a mobile application) from a web API hosted by the company that markets and sells a device model. Mandiant investigated the viability of brute forcing ThroughTek UIDs and found it to be infeasible due to the necessary time and resources.

If an attacker obtains a UID of a victim Kalay device, they can maliciously register a device with the same UID on the network and cause the Kalay servers to overwrite the existing Kalay device. Once an attacker has maliciously registered a UID, any client connection attempts to access the victim UID will be directed to the attacker. The attacker can then continue the connection process and obtain the authentication materials (a username and password) needed to access the device.

With the compromised credentials, an attacker can use the Kalay network to remotely connect to the original device, access AV data, and execute RPC calls. Vulnerabilities [which exist] within the device-implemented RPC interface can lead to fully remote and complete device compromise. Mandiant observed that the binaries on IoT devices processing Kalay data typically ran as the privileged user root and lacked common binary protections such as Address Space Layout Randomization ("ASLR"), Platform Independent Execution ("PIE"), stack canaries, and NX bits.

They then post a video which demonstrates a functional proof of concept for CVE-2021-28372 and they note that they will not be releasing any public exploit code. They conclude by explaining: *"CVE-2021-28372 poses a huge risk to an end user's security and privacy and should be mitigated appropriately. Unprotected devices, such as IoT cameras, can be compromised remotely with access to a UID and further attacks are possible depending on the functionality exposed by a device."*

For my part, I get it that doing all that reverse engineering is their job and that it presents an engaging challenge. But it's a big problem that on the one hand a company like ThroughTek can produce and sell under license an arbitrary protocol and SDK that is closed and proprietary and has undergone no external 3rd party security verification, and also that on the other hand, IoT vendors will purchase licenses to this sort of closed and proprietary technology.

The Mandiant researchers listed a bunch of things that need to be done to tighten up the security of the system. ThroughTek has responded. But those 83 million webcams and baby monitors already sold won't be getting their firmware updated... ever.

Like I said, no one wants that crap sharing the same network as their PCs. When the IoT chickens come home to roost, you want to be a spectator and not a victim.

Miscellany

Overlay Networks

It's clear from the rave reactions from all those who took "TailScale" out for a spin that the era of "Overlay Networks" is upon us. The first popular mainstream overlay network was one that this podcast helped to put on the map. That was Hamachi. And I loved it because back in the pre-IPv4 exhaustion days, it cleverly used the unallocated and never before used class A 5 (dot) network for its overlay endpoint numbering. Another emerging term we're going to be seeing is the notion of "Software Defined Networks" — so, SDN or sometimes SD-WAN.

TailScale: https://tailscale.com/

TailScale appears to have nailed a solution to this need. And, as we've seen, their free tier for personal use has been hugely popular among our listeners. But for the sake of completeness, and because some people will prefer totally open solutions without any commercial component, I wanted to bring three other very similar offerings to everyone's attention:

Nebula: https://github.com/slackhq/nebula

Nebula is a fully open and open source scalable overlay networking tool developed by Slack. It focuses upon performance, simplicity and security to enable its users to seamlessly connect computers anywhere in the world. Nebula is portable. It runs on Linux, OSX, Windows, iOS, and Android. It can be used to connect a small number of computers, but is equally able to connect tens of thousands of computers. Nebula is what Slack runs on.

It incorporates a number of existing concepts like encryption, security groups, certificates, and tunneling. Nebula didn't invent any of those individual pieces. They all existed before Nebula in various forms. What makes Nebula different from existing offerings is that it brings all of these ideas together, resulting in a sum that is greater than its individual parts.

https://slack.engineering/introducing-nebula-the-open-source-global-overlay-network-from-slack/

Slack rhetorically asked: *"What's the easiest way to securely connect tens of thousands of computers, hosted at multiple cloud service providers in dozens of locations around the globe?"* — Slack's answer is Nebula. They wrote: *"At Slack, we asked ourselves this very question a few years ago. We tried a number of approaches to this problem, but each came with trade-offs in performance, security, features, or ease of use. We will gladly share those experiences in future presentations and writing, but for now, just know that we did not set out to write software to solve this problem. Slack is in the business of connecting people, not computers."* What Slack found was that at the time they had no choice other than to build their own solution. So what is Nebula?

Nebula is a mutually authenticated, peer-to-peer, software defined network (SDN) based on the Noise Protocol Framework. (I'll explain about that in a moment.) Nebula uses certificates to assert a node's IP address, name, and membership within user-defined groups. Nebula's user-defined groups allow for provider-agnostic traffic filtering between nodes.

Discovery nodes allow individual peers to find each other and optionally use UDP hole punching to establish connections from behind most firewalls or NATs. Users can move data between nodes in any number of cloud service providers, datacenters, and endpoints, without needing to maintain a particular addressing scheme.

Nebula uses elliptic curve Diffie-Hellman key exchange, and AES-256-GCM in its default configuration. (Precisely the same technologies and protocols I chose for SQRL.)

Nebula was created to provide a mechanism for groups of hosts to communicate securely, even across the Internet, while enabling expressive firewall definitions similar in style to cloud security groups. To set up a Nebula network, you'll need:

1. The Nebula binaries for your specific platform. Specifically, you'll need nebula-cert and the specific nebula binary for each platform you use.

2. (Optional, but you really should..) At least one discovery node with a routable IP address which Nebula calls a "lighthouse". Nebula lighthouses allow nodes to find each other, anywhere in the world. A lighthouse is the only node in a Nebula network whose IP should not change. Running a lighthouse requires very few compute resources, and you can easily use the least expensive option from a cloud hosting provider. If you're not sure which provider to use, Slack wrote that a number of them have used $5/mo DigitalOcean droplets as lighthouses.

Once a Nebula instance has been launched, ensure that Nebula UDP traffic (default port udp/4242) can reach it over the internet.

The Noise Protocol Framework: Nebula uses a term we've not talked about before — The Noise Protocol Framework. https://noiseprotocol.org/ The Noise Protocol Framework describes itself: "Noise is a framework for building crypto protocols. Noise protocols support mutual and optional authentication, identity hiding, forward secrecy, zero round-trip encryption, and other advanced

features." Open source implementations are available in C, C#, Go, Haskell, Java, Javascript, Python, and Rust. Noise is currently used by WhatsApp and WireGuard, and others.

In other words, rather building a solution on top of WireGuard as some of the other Software Defined overlay networks do, essentially creating a dynamic configuration manager for WireGuard — which is an entirely acceptable solution by the way — Nebula drops down to a lower level to use the same Noise Protocol Framework upon which WireGuard was built. So there is no separate instance of WireGuard and Nebula incorporates the features and benefits provided by WireGuard by virtue of being based upon the same shared Framework.

Another open source solution, known as Innernet, IS built upon WireGuard:

Innernet: https://github.com/tonarino/innernet

The Innernet developers explain: "Innernet is similar in its goals to Slack's Nebula or Tailscale, but takes a bit of a different approach. It aims to take advantage of existing networking concepts like CIDRs (classless inter-domain routing) and the security properties of WireGuard, to turn your computer's basic IP networking into more powerful ACL (access control) primitives. This allows you to nurture and shape your own private networks with simple, free, open-source infrastructure: https://blog.tonari.no/introducing-innernet

We had some simple goals:

- Conveniences a typical WireGuard user wants: peer names, auto-updating peer lists, groups based on IP blocks, and automatic NAT holepunching.
- Free, open source, and made to be self-hosted. We think it's especially important for such a vital and low-level piece of our infrastructure to not be dependent on the livelihood of a company one has no control over.
- Straightforward architecture — no Raft consensus here. It's a simple SQLite server/client model.

And finally, one last commercial offering known as "ZeroTier":

ZeroTier: https://www.zerotier.com/pricing/

Their slogan is "It Just Works" ZeroTier combines the capabilities of VPN and an SD-WAN to simplify network management. Enjoy flexibility while avoiding costly hardware vendor lock in.

- SPEED: Set up ZeroTier in minutes with remote, automated deployment.
- FLEXIBILITY: Emulates Layer 2 Ethernet (IP level) with multipath, multicast, and bridging capabilities.
- SECURITY: ZeroTier's zero-trust networking solution provides scalable security with 256-bit end-to-end encryption.
- It is FREE for up to 50 endpoints. (So if TailScale's 20-node free limit might have been a bit binding, ZeroTier is another.)

I poked around a bit to get some feeling for ZeroTier. It is all open, open source and fully documented. And it's possible to self-host. And they provide a free, global, DNS-like directory service for all users. They rolled their own crypto solution, though they did use all of the latest crypto primitives. So long as a system's design is kept clean, simple and straightforward it's no longer difficult to create a provably secure solution. It's when designers start getting fancy and tricky that edge cases creep in.

So in summary... **Innernet** looks like a free and open source solution providing much of what **Tailscale** offers, both being built on top of WireGuard to give WireGuard a bunch of needed features. **Nebula** is a free and open source SDN (software defined network) written from scratch by the Slack guys because nothing that existed at the time did the job they needed; and **Nebula** is built using the same crypto framework as WireGuard thus inheriting many of the same guarantees as WireGuard. And by forming the backbone of Slack, Nebula has already been proven to be scalable to global scale — not that any of us will need that, but it's kinda cool.

Closing the Loop

Bryan notes in GRC's Security Now newsgroup:
One thing Steve didn't mention is that in 2016 Avast also purchased AVG for 1.3 billion. So now Norton Gobbled up, Avera, Avast and AVG...

I'm glad Bryan mentioned that. It happened on our watch in July of 2016. Avast purchased its fellow Czech cybersecurity company AVG for $1.3 billion. So we are seeing a clear merging and consolidation of the independent security add-on companies.

A DM'er (via DM, so I didn't have permission to share his identity)
"Can I ask for an advice? Where do you see the future in IT? I have been working for the last 8 years in different position mainly in IT infrastructure (Network and System) but now I want to invest my time and plan my future. Your podcast is a great library for understanding Security. I want to start from the bottom and learn my way up."

Microsoft's Reasoned Neglect

Not long after we finished recording last week's podcast which, as we all know, was titled "Microsoft's Culpable Negligence" I had another thought. Last week I said over and over, and drove the point home, that given Microsoft's effectively unlimited resources, and having clear knowledge of new highly critical vulnerabilities handed to them — and of the devastating impact the exploitation of those vulnerabilities would have — I was unable to see any rational explanation for Microsoft's behavior, other than that they had to be performing a brutal cost-benefit analysis and rationally deciding not to fix those vulnerabilities. One way or another, for one reason or another, this **had** to be a deliberate decision, because there was **no way** to parse the history, that we have all been witness to, that wouldn't cause any unbiased observer to conclude that what has been happening was exactly what Microsoft had **decided** should be happening — insane as that at first appears.

Last week, I stated that there could be no other reason. Then I thought of one.

There **is** an explanation that perfectly maps onto all of the evidence, and exactly **predicts** the behavior we are all witnessing from Microsoft. And in this proposed model, the driving motivation is, indeed, a brutal cost-benefit analysis—but one that's even more brutal than we imagined. It's just not the obvious cost-benefit analysis I was focused upon and described last week.

Last week I was assuming that it would only be **hostile and malicious adversaries** who would be attacking users of Microsoft's software. Thus, "the cost" in the cost-benefit analysis would be the attacks themselves. But what if, instead, "attacks" were the **benefits**, and what if those benefits arising from attacks were so beneficial that they outweighed the cost to Microsoft, which we've already determined to be effectively negligible?

So then we have to ask: How could attacks on users of Microsoft's proprietary software be beneficial? Such attacks would be in the U.S. national interest if they were being conducted **by** the United States domestic intelligence services **against** U.S. foreign adversaries.

I recall mentioning on this podcast, many years ago, that Microsoft routinely tipped-off our U.S. intelligence agencies about recently discovered and not-yet-patched flaws in Windows and in their various other products.

On Security Now! Episode #426, which we recorded on October 16th, 2013, I quoted Bruce Schneier from a piece he wrote for The Atlantic titled "How the NSA Thinks About Secrecy and Risk." https://www.theatlantic.com/technology/archive/2013/10/how-the-nsa-thinks-about-secrecy-and-risk/280258/

I'm going to read directly, verbatim, the first five paragraphs of that piece, which Bruce wrote nearly eight years ago:

As I report in The Guardian today, the NSA has secret servers on the Internet that hack into other computers, codename FOXACID. These servers provide an excellent demonstration of how the NSA approaches risk management, and exposes flaws in how the agency thinks about the secrecy of its own programs.

Here are the FOXACID basics: By the time the NSA tricks a target into visiting one of those servers, it already knows exactly who that target is, who wants him eavesdropped on, and the expected value of the data it hopes to receive. Based on that information, the server can automatically decide what exploit to serve the target, taking into account the risks associated with attacking the target, as well as the benefits of a successful attack. According to a top-secret operational procedures manual provided by Edward Snowden, an exploit named Validator might be the default, but the NSA has a variety of options. The documentation mentions United Rake, Peddle Cheap, Packet Wrench, and Beach Head—all delivered from a FOXACID subsystem called Ferret Cannon. Oh how I love some of these code names. (On the other hand, EGOTISTICALGIRAFFE has to be the dumbest code name ever.)

Snowden explained this to Guardian reporter Glenn Greenwald in Hong Kong. If the target is a high-value one, FOXACID might run a rare zero-day exploit that it developed or purchased. If the target is technically sophisticated, FOXACID might decide that there's too much chance for discovery, and keeping the zero-day exploit a secret is more important. If the target is a low-value one, FOXACID might run an exploit that's less valuable. If the target is low-value and technically sophisticated, FOXACID might even run an already-known vulnerability.

***We know that the NSA receives advance warning from Microsoft of vulnerabilities that will soon be patched**; there's not much of a loss if an exploit based on that vulnerability is discovered. FOXACID has tiers of exploits it can run, and uses a complicated trade-off system to determine which one to run against any particular target.*

This cost-benefit analysis doesn't end at successful exploitation. According to Snowden, the TAO—that's Tailored Access Operations—operators running the FOXACID system have a detailed flowchart, with tons of rules about when to stop. If something doesn't work, stop. If they detect a PSP, a personal security product, stop. If anything goes weird, stop. This is how the NSA avoids detection, and also how it takes mid-level computer operators and turn them into what they call "cyberwarriors." It's not that they're skilled hackers, it's that the procedures do the work for them.

Okay. So in the 4th paragraph of that longer piece, famous security expert Bruce Schneier said:

> *"We know that the NSA receives advance warning from Microsoft of vulnerabilities that will soon be patched; there's not much of a loss if an exploit based on that vulnerability is discovered."*

The revelations made by Edward Snowden and Wikileaks stripped us of our innocence and matured our understanding of the true nature of the global cyber-intelligence world. Sometimes the need to gather intelligence requires, how shall I put it? **...an extreme lack of passivity.**

So, once again I'm being entirely serious about this. Think about it for a moment. Microsoft receives notification of a critical vulnerability from any of the world's many white-hat hackers who are poking and prodding their products. Say... of a horrifically exploitable flaw in their eMail Exchange Server.

The exploit is not publicly known and its discoverer has agreed to keep it to themselves until sometime after it has been fixed. So here's how this proposed timeline would play out...

1. Microsoft thanks the security researching hacker and promises to graciously throw them a bone by mentioning his discovery in their eventual disclosure. Perhaps they'll also receive a bug bounty... but, of course, only if they remain silent until the problem has been fixed and a sufficient number of systems have been patched.

2. Microsoft then uses their well-established quiet backchannel to pass the researcher's findings on to the NSA and the CIA. Microsoft takes no active part in the development of an exploit, because that would be crossing a line. And, should it ever become common knowledge that this early information was provided to U.S. intelligence services, Microsoft is simply being a good citizen and helping our own domestic intelligence services to guard against attacks which might exploit this now-discovered yet not yet publicly disclosed vulnerability.

3. And now... Microsoft sits on it. Remember how Victor Mata, who reported a remote code execution vulnerability with full SYSTEM privileges, tweeted Wednesday before last, on August 11th *"Hey guys, I reported the vulnerability in December 2020, but haven't disclosed details at MSRC's request. It looks like they acknowledged it today due to the recent events with print spooler."* So, when Microsoft finds themselves in receipt of a valuable vulnerability do they immediately assign it a CVS number? No. That would raise suspicion and speculation and might start people looking. Do they post a note about a new vulnerability that needs fixing and their intention to do so? Nope. Not only do they remain completely mum, **they do not patch it.** It is only of use to our own, shall we say, "proactive" intelligence agencies who have been informed of it, so long as it remains unknown and unpatched.

   All evidence suggests that the reasoning here is that as long as it has not been discovered and publicly reported as being actively exploited in the wild, it's just like any of all those other undiscovered vulnerabilities that exist within Windows—and we all know there's certainly no shortage of those. In the juicy case of Microsoft's Exchange Server, it's been sitting there, previously undiscovered for more than the past 10 years. So what's a few more months? And just **think** of all the benefit that our domestic intelligence agencies can reap from it until, and if, it eventually comes to light on its own! Or perhaps, as Bruce observes, as an unfortunate side effect of its active use by our own NSA or CIA? What was it that we heard about the Exchange Server exploit? That it could allow an adversary access to all of the server's local communication history? Think that might be of interest to some of our snoopier spooks?

4. And, finally, consider that it wasn't until the ProxyLogon vulnerabilities were suddenly found to be exploited by adversaries actively attacking users of Exchange Server that Microsoft finally jumped to attention and rushed out an emergency fix for the problem. Remember that they claimed to be getting ready to release the fix with the next Patch Tuesday? Uhhh Huh. And remember that something didn't smell quite right about that at the time? And then, since this sudden disclosure apparently caught them by surprise, and they were unable to deny how long ago they had been originally informed about it — by our friend Orange Tsai! — they then hinted that someone they told might have leaked the details. Right. More like someone they told might have been caught privately exploiting it.

I frequently hear that we have listeners who have been with us from the start—for all of the past 16 years. So those people will have heard every podcast we've produced and they will know that they've never once heard me jump onto a conspiracy theory. That's not the way I roll, and I certainly have no intention of doing so now. I have no firsthand evidence of this, and I'm not particularly interested in digging up any. As we all know, I have many much better things to do.

But we also know that absence of proof is not proof of absence. And any system such as this would be well designed to remain off the books and off the radar.

The chart above was made on March 24th, several weeks **after** Microsoft's emergency patch release for the ProxyLogon flaws. In it, we see 2,496 still-vulnerable Exchange Servers in Russia and 1,473 Servers still vulnerable in China. Again... weeks downstream. I wonder how many Exchange Servers in Russia and China might have been vulnerable before Microsoft's revelation?

If nothing else, this resolves the apparent mystery of **Microsoft's culpable negligence** by converting apparent negligence into **reasoned neglect**. I guess the question remains... which would we rather have? A negligent Microsoft or a diabolically neglectful Microsoft? Neither place the interests of their own customer's first.