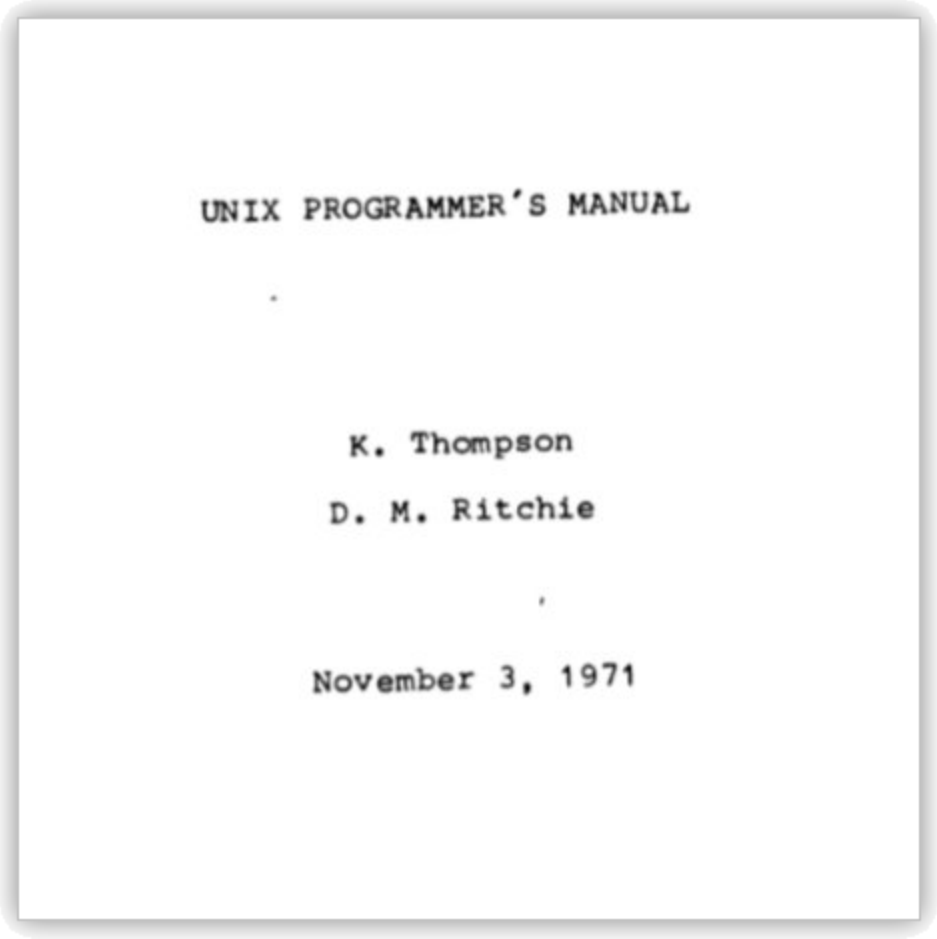
# Security Now! #844 - 11-09-21

# Bluetooth Fingerprinting

## This week on Security Now!

This week we quickly cover a bunch of welcome news on the combating ransomware front. We look at the results from last week's Pwn2Own contest in Austin Texas and at a weird problem that only some users of Windows 11 started experiencing after Halloween. There's a serious problem with GitLab servers and additional supply-chain attacks on JavaScript's package management. Google fixed a bunch of things in Android last Tuesday, and Cisco has issued an emergency CVSS 9.8 alert and US Federal agencies are being ordered to patch hundreds of outstanding vulnerabilities. We have some fun closing the loop feedback from our listeners. I'm going to share the details of an interesting IRQ problem I tracked down last week. Then we'll take a look at an aspect of radio frequency fingerprinting that has apparently escaped everyone's notice until seven researchers from UCSD did the math.

## November 3rd, 2021 was Unix's 50th Anniversary


UNIX PROGRAMMER'S MANUAL

K. Thompson

D. M. Ritchie

November 3, 1971

*The 50th Anniversary of the 1st Edition of Unix. It ran on a PDP-11/20 with 8K words (16,384 bytes) or RAM… with room for programs.*

# Ransomware News

**Lots of welcome progress on the ransomware front**

Much of the past week's security news were reports of the success of the various counter-ransomware campaigns now being actively waged. Yesterday, the US Department of Justice charged a 22-year-old Ukrainian national for orchestrating the ransomware attacks which were facilitated by flaws in Kaseya's servers. Following an arrest warrant issued by the US, the suspect, Yaroslav Vasinskyi, was detained last month by Polish authorities at a border station while crossing from Ukraine into Poland. In court documents unsealed yesterday, the DOJ said that Vasinskyi was a long-time collaborator of REvil group.

The US has also charged a second suspect that helped the REvil gang deploy its ransomware. Identified in court documents as 28-year-old, Yevgeniy Polyanin, the DOJ said this Russian national also worked as a REvil affiliate. US officials believe that Polyanin is the person who breached the network of TSM Consulting, a Texas-based managed service provider, from where he deployed the REvil ransomware on the internal networks of at least 20 Texas local government agencies on August 16, 2019. Though Polyanin is still at large and wanted by the FBI, the DOJ said it managed to successfully seize $6.1 million worth of cryptocurrency assets that the suspect was holding in an FTX account.

The US announcements yesterday came just hours after Europol announced similar arrests in Romania, Kuwait, and South Korea. Seven members of affiliate groups who worked with the GandCrab and REvil ransom-as-a-service programs have been detained.

And yesterday, the US Treasury Department imposed sanctions on the cryptocurrency exchange Chatex for facilitating financial transactions for ransomware actors. An analysis of Chatex's known transactions indicated that over half were directly traced to illicit or high-risk activities such as darknet markets, high-risk exchanges, and ransomware. Officials said that Chatex also had direct ties to Suex, a Russian cryptocurrency exchange portal which the Treasury sanctioned in September for the same reasons. Treasury also sanctioned three Chatex suppliers: IZIBITS OU, Chatextech SIA, and Hightrade Finance Ltd identifying them as three companies set up as infrastructure for Chatex, enabling Chatex's operation. Operations for Chatextech and IZIBITS have been suspended by officials from Latvia and Estonia, respectively. Latvian officials are currently working to identify Chatex board owners, all non-Latvian nationals.

And to encourage more arrests, the US State Department announced bounties for information that may lead to the identification and/or arrest of members of the REvil / Sodinokibi ransomware group: $10 million for information on REvil key leaders and $5 million for information on REvil affiliates. This follows last Thursday's identical reward announcements for any information which may lead to the identification and/or arrest of members of the Darkside ransomware group.

And the last piece of welcome news from the ransomware universe is that the BlackMatter ransomware-as-a-service group announced last week that it was shutting down its operations as a result of pressure from authorities. Better to slink off in the night than be dragged away in handcuffs.

So, it seems pretty clear that the US and its global partners are seriously turning up the heat on the ransomware business. As a result, it's much less clear today that hacking and extorting is an easy and safe way to make a buck.

## Security News

**Pwn2Own Austin: Last Tuesday-Thursday largest ever 3-day Fall 2021 Pwn2Own.** Since its inception, the Fall Pwn2Own contest has focused on consumer devices while the contest location has wandered around the globe. Nine years ago, the 2012 Amsterdam Pwn2Own targeted only mobile phones. In the years that followed the context grew to include smart TVs, wearables, and smart speakers. Last year's contest in Toronto was further expanded to include Network Attached Storage (NAS) devices. And this year's 2021 contest was held in Austin, Texas with further growth in home-office equipment expanding the router category and adding a printer category. In all, 22 devices were available as targets with more than $500,000 USD in prize money.

In the past, we've had fun recreating a running commentary of the event. But as I scanned through it, it was clear that the contest has grown so large that doing that again would take up half of the podcast. So I'm going to mention the equipment under attack, then hit the high achievement points from the 3-day event.

Pwn2Own has its root in mobile handsets, so that category remains well represented with the Google Pixel 5, the Samsung Galaxy S21, and Apple's iPhone 12. Each handset will be running the latest version of their respective operating systems with all available updates installed. And the context has increased the rewards for those three targets to increase the incentives for cracking those handsets.

As we know, this was the year of print spooler bugs. But the competition wanted to see about the devices themselves. So three printer from HP, Lexmark, and Canon will be on the chopping block: the HP Color LaserJet Pro MFP M283fdw, Lexmark's MC3224i, and Canon's ImageCLASS MF644Cdw.

In the home automation category we have the Facebook Portal, Amazon's Echo Show 10, Google's Nest Hub (2nd Gen), the Sono One Speaker and Apple's HomePod mini. Two smart TVs, the Sony X80J Series - 43" and the Samsung Q60A Series – 43" will both be put to the test. With all the problems we've seen from router compromises, putting five routers up for attack makes sense. So available targets are the TP-Link AC1750 Smart Wi-Fi Router, NetGear's Nighthawk Smart Wi-Fi Router (R6700 AC1750), Cisco's RV340, Mikrotik's RB4011iGS+RM and Ubiquiti Network's EdgeRouter 4. Representing the network attached storage category we have Synology's DiskStation DS920+, Western Digital's My Cloud Pro Series PR4100 and also WD's 3TB My Cloud Home Personal Cloud. And last but not least, the single entry in the external storage category is SanDisk's Professional G-DRIVE ArmorLock SSD 1TB.

So what happened?

The super-abbreviated version, which I'll expand upon in a moment, is that the uber-talented participants in this year's event revealed for the first time ever their discoveries of 61 brand new exploitable vulnerabilities across that range of fully-patched commercial products and the participants collectively took home $1,081,250 for the second Pwn2Own in a row to clear the $1 million mark.

And as for the success of the individual participants and teams, recall that the way this works is that awards take the form of cash prizes and Master of Pwn points which are totaled to determine each year's Master of Pwn winner.

The French offensive security firm, Synacktiv, topped the 3-day contest's leaderboard by winning 20 Master of Pwn points and earning themselves $197,500 by netting maximum points for a 0-day vulnerability in the Sonos One smart speaker. They successfully demonstrated seizing full control of the Sonos One via a stack-based buffer overflow flaw accounted for six points and $60,000 of their winnings. They also scored another 4 points and $40,000 from leveraging a configuration flaw which gave them code execution on Western Digital's My Cloud Pro Series PR4100 NAS.

Trailing Synacktiv in second place by only 2 points were joint winners of the flagship Spring event DEVCORE, who earned 18 points and $180,000 in total. Together with his fellow DEVCORE members, Orange Tsai – who, as we'll recall, was the hacker to discover "a whole new attack surface" on Microsoft Exchange Server last year – also claimed maximum points for compromising Sonos One, along with four points and $40,000 after combining out-of-bounds read and out-of-bounds write flaws to hack Western Digital's 3TB My Cloud Home Personal Cloud.

STARLabs, which finished third overall, chained out-of-bounds read with heap-based buffer overflow bugs on the beta version of the same device, earning five points and $45,000.

Fourth on the final standings was Sam Thomas from the UK infosec firm Pentest Ltd. Sam earned $40,000 and 4 points after chaining three bugs to get code execution on WD's PR4100.

I have a link to the detailed blow-by-blow rundown in the show notes. It's really quite something: https://www.zerodayinitiative.com/blog/2021/11/1/pwn2ownaustin

For example, one researcher named Bien Pham (@bienpnn) from Team Orca of Sea Security really had his way with the routers he attacked. In his first event, he leveraged a logic error to compromise the WAN interface of the Cisco RV340 router to win himself $30,000 and 3 Master of Pwn points. Then, two hours later he used a three-bug chain, including an auth bypass and a command injection, to take over the LAN interface of that same Cisco RV340 router to earn an additional $15,000 and 2 more Master of Pwn points. And finally, he finished the first day by using an OOB Read bug to take control of the TP-Link AC1750 router via the LAN interface for $5,000 and 1 Master of Pwn point.

Many of the other individuals and teams demonstrated similar quite impressive (or horrifying, depending upon your position) skill. And in the process they successfully demonstrated the exploitation of 61 new previously unknown vulnerabilities — about twice the previous record.

As always, the participants immediately provide full disclosures to the affected vendors and will withhold their public disclosure for 120 days, after which, patched or not, they're free to disclose the technical details of their wizardry.

After the competition, Dustin Childs, the communications manager for the Zero Day Initiative contest holder, was asked to name his favorite exploit. He replied: "It's hard to beat an exploit that turns a printer into a jukebox and plays AC/DC." Dustin was referring to the impressive work of the three man F-Secure Labs team who targeted the HP Color LaserJet Pro MFP M283fdw... to make it play music — or latent least make a lot of noise depending upon how you feel about AC/DC.

**Windows 11 snipping tool, its emoji picker, and other parts are failing**

The Verge carried an intriguing report last Thursday that some, apparently quite selective, parts of Windows 11 were failing after Halloween (October 31st, so exactly at the end of a month). The Verge's headline provides our first clue: "Microsoft warns Windows 11 features are failing due to its expired certificate" But it's unclear what "its" refers to here. The Verge wrote:

Microsoft has started warning Windows 11 users that certain features in the operating system are failing to load due to an expired certificate. The certificate expired on October 31st, and Microsoft warns that some Windows 11 users aren't able to open apps like the Snipping Tool, touch keyboard, or emoji panel.

A patch is available to fix some of the issues, but it's currently in preview, meaning you have to install it manually from Windows Update. The patch, KB4006746, will fix the touch keyboard, voice typing, emoji panel, and issues with the getting started and tips sections of Windows 11. You'll be able to find this patch by checking for updates in the Windows Update section of Settings in Windows 11.

Microsoft's patch doesn't address the problems with the Snipping Tool app, though. Microsoft says: "To mitigate the issue with Snipping Tool, use the Print Screen key on your keyboard and paste the screenshot into your document. You can also paste it into Paint to select and copy the section you want." So, in other words, don't use the Snipping Tool.

It's not clear how many Windows 11 users are affected by these issues, and we haven't been able to replicate the Snipping Tool problems on multiple patched systems. If you are having issues, some Verge readers have reported being able to change the system date back to October 30th, and then launching Snipping Tool to get it working again. You can then change the system date back once the app has loaded ok.

The expired certificate is also causing issues with the accounts page in the settings section of Windows 11 with S mode enabled and the input method editor UI. It's not clear when the Snipping Tool and S mode issues will be addressed. Microsoft says: "We are working on a resolution for Snipping Tool and the S mode only issues and will provide an update when more information is available."

If we reverse engineer the trouble, it sounds as though some Windows 11 apps have, themselves, been signed in such a way that their own digital signatures will not validate as of

November 1st, 2021. But the way Authenticode signatures are designed, if the signature was valid at the time of signing, the signatures themselves never expire. So this cannot be the signature itself that's expiring. But, like any digital signature, it's based upon a chain of trust which typically has an intermediate certificate and a root certificate. So, if any certificate in the chain being used by any of those apps expired, even though the apps' signing certificate itself was still technically valid, Windows would not be able to validate its signature and would refuse to run.

Now, as to why only some Windows 11 users are experiencing this, and why among those who are, only some of their apps are seeing this, and why a fix was quickly deployed for most of the apps but not for the Snipping Tool, this is additional evidence that, for whatever reason, Windows is growing ever more complex. I use and love Windows; so I dearly hope they're able to hold it together.

**Trouble being created by unpatched GitLab servers**

So, here's the sales pitch for GitLab servers...

A page on TechRepublic introduces the reason for wanting one as follow:

> *If you're a Git user, you know that having local repositories that can be accessed via a local LAN (or external WAN) is a crucial element of the development process. You can certainly opt to go with GitHub, but that negates the ability to host locally. So when you want to host your own repositories, where do you turn? In a word, GitLab. GitLab allows you to host an on-premise Git repository that can be accessed from either your local LAN or (if you have an available public IP address) from outside your company. GitLab is fairly easy to install and incredibly simple to use.*

Right... and unfortunately unpatched GitLab servers also contain a now widely exploited flaw that's been used to create multi-thousand member botnets which are generating in excess of one terabit per second of DDoS attack traffic. Why? Because they're not only easy-to-deploy but "wouldn't it be just great to put this on the WAN!?"

So here's the situation:

Bad guys are exploiting a security flaw in these GitLab self-hosted servers to assemble botnets and launch gigantic DDoS attacks — some in excess of a terabit/second.

Damian Menscher, a Security Reliability Engineer at Google Cloud, responsible for Google's DDoS defenses, disclosed last Thursday that attackers are exploiting CVE-2021-22205, a vulnerability that GitLab **patched** back in April of this year. But of the 60,000 GitLab servers publicly exposed to the Internet, only about half have since been updated with the patch.

The flaw exists in GitLab's ExifTool, a library that's commonly used to remove the metadata from images uploaded to web servers, was discovered by William Bowling and reported to GitLab via its bug bounty program over at HackerOne. In a report filed via HackerOne, Bowling said he discovered a way to abuse how ExifTool handles uploads for DjVu file format used for scanned documents to gain control over the entire underlying GitLab web server.

Public proof-of-concept code for this vulnerability appeared in June, around the same time that the Italian security firm HN Security first spotted attacks. At the time, an HN security researcher said the company began an investigation after spotting randomly-named user accounts being added to compromised GitLab servers... user accounts that were most likely created by the attackers to allow remote control of the hacked systems.

While the purpose of these intrusions remained unclear to HN Security, yesterday, Google's Menscher said the hacked servers were part of a botnet comprising of "thousands of compromised GitLab instances" that was launching large-scale DDoS attacks.

As we have so often observed, botnet operators are exploiting the tardiness of individuals and companies around the world when it comes to patching their software, in this case, in-house GitLab servers.

According to a Rapid7 analysis published last Monday, there are more than 60,000 GitLab servers connected to the internet, of which around half still remain unpatched for the CVE-2021-22205 ExifTool exploit.

And it's worth noting that GitLab is not the only user of the ExitTool. So the exploitation of its flaw could very well also impact other types of web applications where the tool might be deployed. So it may be that additional exploitation will be forthcoming and that other types of web apps might need patching as well.

One way to prevent attacks might be to prevent uploading of DjVu files at the server if companies don't need to handle this file type. On the other hand, if you're going to go to that trouble, updating GitLab with all of the current patches should be done first.

**More supply chain attacks**

Successful attacks on the software supply chain continue to be discovered. Which, of course, begs the question, which unknown attacks are currently in place that haven't yet been detected?

In this most recent instance of discovery, a pair of extremely popular JavaScript npm packages, "coa" and "rc", which boast combined weekly downloads of 23 million, were discovered to have been infected with password-stealing malware. The security team responsible for the npm JavaScript package manager has warned users that two of its most popular packages had been hijacked by a threat actor.

As I said, the two affected packages are "coa" and "rc". Coa is a command-line argument parser with ~8.8 million weekly downloads and "rc" is a configuration loader with ~14.2 million weekly downloads.

Compromised coa versions: 2.0.3, 2.0.4, 2.1.1, 2.1.3, 3.0.1, 3.1.3
Compromised rc versions: 1.2.9, 1.3.9, 2.3.9.

Both packages appear to have been compromised at the same time and were the result of attackers gaining access to a package developer's account. Once inside, the threat actor added a

post-installation script to the original codebase. The post-installation ran an obfuscated TypeScript, that would check for operating system details and download either a Windows batch or Linux bash script. The deobfuscated version of the Windows batch script revealed that in the case of Windows, the compromised packages would download and run a DLL file contained a version of the Qakbot trojan. What a mess.

The compromise to "coa" was spotted first after its new installation routine started crashing build pipelines for React-based applications. Last Thursday, the npm team tweeted shortly after detecting the coa compromise following a wave of reports about failed builds:

*"The compromised [developer] account has been temporarily disabled and we are actively investigating the incident and monitoring for similar activity. We will share additional information as appropriate based on our investigation."* — npm (@npmjs) November 4, 2021

The matching compromise to the "rc" package was discovered a few hours later.

The npm security team removed all the compromised coa and rc versions to prevent developers from accidentally infecting themselves. But it appears unlikely that either of the compromises had much chance of slipping through. Builds using it were crashing. Both libraries are widely used. The malicious code was not well hidden, and both libraries hadn't seen any new releases since December 2018 and December 2015, respectively. So that alone would have raised sufficient suspicion to trigger a security audit within most professional developer teams.

And one last interesting point: The malicious code present in these incidents is nearly identical to the code used in the compromise of the User-Agent parser (UAParser) library which occurred late last month, and which we covered here at the time. So it appears that someone, and apparently someone either not highly skilled or someone in a hurry, has set their sights upon JavaScript and the npm supply chain for exploitation.

**If it's Tuesday...**

Today is Patch Tuesday for Microsoft as well as for many others in the industry. So, next week we'll check back to see what new adventures in computing have been created by today's updates.

But last Tuesday Google rolled out its monthly security patches for Android, fixing 39 flaws including a 0-day that it said was being actively exploited in the wild in limited, targeted attacks.

That was is being tracked as CVE-2021-1048, a use-after-free vulnerability in the kernel that can be exploited to obtain local privilege escalation. Since we depend upon code containment to a great degree in today computing, although privilege escalation is not as scary sounding as remote code execution, it's definitely important. And the very fact that this flaw was being used in the wild demonstrates that it was doubtless quite useful for some nefarious purpose.

Aside from that 0-day, last Tuesday's patches also foreclosed on a pair of critical remote code execution (RCE) vulnerabilities in the System component that could allow remote adversaries to execute malicious code within the context of a privileged process by sending a specially-crafted transmission to targeted devices.

Two other critical flaws in Qualcomm's closed-source components were also fixed along with a 5th critical vulnerability in Android TV which could have permited an attacker in close proximity to silently pair with a TV and execute arbitrary code with no privileges or user interaction required.

In terms of actively exploited, in-the-wild total 0-days found and fixed so far this year, when compared with Windows, Chrome and even iOS, Android has been faring surprisingly well. After counting last Tuesdays latest addition, Android has only needed to address a total of 6 0-days this year. And it's not as if there isn't tremendous pressure to pry into Android. So, way to go, Google!

**Cisco's DEFAULT SSH key**

Last Thursday, Cisco released an update to fix one of those rare CVSS 9.8 vulnerabilities in their Policy Suite products.
https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-cps-static-key-JmS92hNv

However, the announcement of this is somewhat misleading because they say:

> *A vulnerability in the key-based SSH authentication mechanism of Cisco Policy Suite could allow an unauthenticated, remote attacker to log in to an affected system as the root user.*
>
> *This vulnerability is due to a weakness in the SSH subsystem of an affected system. An attacker could exploit this vulnerability by connecting to an affected device through SSH. A successful exploit could allow the attacker to log in to an affected system as the root user.*
>
> *Cisco has released software updates that address this vulnerability. There are no workarounds that address this vulnerability.*

Now... you don't get to say that "A vulnerability in the key-based SSH authentication mechanism could allow an unauthenticated, remote attacker to log in to an affected system as the root user." when the vulnerability in question is a hard-coded pre-set factory default SSH key which is readily discoverable in the device's firmware.

So, is this vulnerability, as they said "due to a weakness in the SSH subsystem of an affected system." Uhhh... maybe. It would be a weakness in the SSH subsystem that it contained a default key. The good news is, this was not discovered being used in the wild. To their credit, Cisco discovered this problem on their own. So I want to give them props for looking at their own code, presumably being horrified by what they found, and addressing the problem. After updating their devices, the new installation process create unique SSH keys on the fly rather than shipping every device with the same default starter key.

But this keeps happening to Cisco; we've talked about many similar default login credential problems in the past; the bigger and more important question is how this horribly weak design pattern ever became policy in the first place. We know that the awareness of security has been growing over time. But was there ever a time when shipping enterprise class networking

equipment with factory preset SSH credentials would have been reasonable?

The other worry is that the previous instances of Cisco's default credentials were quite a while ago. How is it that the discovery and remediation of those, several years ago, would not have triggered a full product offering audit and that only now, years later, another similar problem is being found?

**U.S. Federal agencies have been ordered to patch hundreds of actively exploited flaws**

One way to fight the threat of foreign cyberattacks such as ransomware is to go after the individuals behind those attacks. But the other way is to look inward and examine how those attacks are being successful in the first place. So, to that end, CISA, our U.S. Cybersecurity and Infrastructure Security Agency, last week published a catalog of known, exploited and patched vulnerabilities, including those from Apple, Cisco, Microsoft, and Google. And in addition to this catalog, CISA has issued an ORDER requiring all federal agencies to prioritize applying patches for those security flaws within "aggressive" timeframes. I'll say it's aggressive!

The "Binding Operational Directive" (BOD) issued last Wednesday said: *"These vulnerabilities pose significant risk to agencies and the federal enterprise. It is essential to aggressively remediate known exploited vulnerabilities to protect federal information systems and reduce cyber incidents."*

The catalog lists around 176 vulnerabilities identified between 2017 and 2020, and 100 flaws from 2021 alone. And the catalog will be updated with additional actively exploited vulnerabilities as and when they become known provided they have been assigned Common Vulnerabilities and Exposures (CVE) identifiers and have clear remediation action.

The binding directive mandates that security vulnerabilities discovered this year, in 2021 must be addressed by next Wednesday, November 17, 2021, while setting a patching deadline of May 3, 2022 for the remaining older vulnerabilities. Although the Binding Operational Directive is primarily aimed at federal civilian agencies, CISA is recommending that private businesses and state entities review the catalog and remediate the vulnerabilities to strengthen their security and resilience posture.

The new strategy also sees the agency moving away from strictly severity-based vulnerability remediation to those that pose significant risk and are being abused in real-world intrusions in light of the fact that adversaries do not only use "critical" weaknesses to achieve their goals. Some of the most widespread and devastating attacks have chained multiple vulnerabilities rated only high, medium or even low.

Tim Erlin, Tripwire's VP of Strategy, said: *"This directive does two things. First, it establishes an agreed upon list of vulnerabilities that are being actively exploited. Second, it provides due dates for remediating those vulnerabilities. By providing a common list of vulnerabilities to target for remediation, CISA is effectively leveling the playing field for agencies in terms of prioritization. It's no longer up to each individual agency to decide which vulnerabilities are the highest priority to patch."*

And I'll add that it will also, presumably, give federal agency IT departments some needed leverage over their own management who might otherwise not provide them with the time, talent, and budget required to meet the mandate's deadline. Now the IT department can say "You want us to meet this order's deadline? Here's what that's going to take..."

## Closing The Loop

Twitter was the conduit throughout all of the past week for showing me how much this podcast's listeners have grown to know me. A barrage of welcome tweets all said the same thing.

I lost count of the number of people who wrote very close variations of *"I can just hear you saying: What could possibly go wrong."* and then they attached links to the tech news coverage that during last week's Ignite conference Microsoft announced that Excel would now be supporting JavaScript. So yes, indeed... *"What — could possibly go wrong?"* It seems like a great idea for keeping this podcast going for another 17 years.

And then we had the meta-tweet from **Joel Pomales / @joelpomales** who wrote:
Pretty sure @SGgrc is getting bombarded with "What could possibly go wrong?" tweets!

And in another example of knowing me so well, **Kevin Jones / @kevinrj** tweeted: I thought @SGgrc would appreciate this. When Kevin attaches a tweet from "I Am Developer":

> **I Am Devloper @iamdevloper**
>
> 1969:
> -what're you doing with that 2KB of RAM?
> -sending people to the moon
>
> 2017:
> -what're you doing with that 1.5GB of RAM?
> -running Slack

Huh. Yeah... that's progress for ya.

And following up on last week's "Trojan Source" topic, **Joseph Lee / @rippledj** tweeted: @SGgrc GitHub now has a warning about hidden Unicode directional control characters for source code:
https://github.blog/changelog/2021-10-31-warning-about-bidirectional-unicode-text/

> The GitHub Changelog for Halloween: October 31, 2021
>
> Warning about bidirectional Unicode text
> A warning is now displayed when a file's contents include bidirectional Unicode text. Such text can be interpreted or compiled differently than it appears in a user interface. For example, hidden, bidirectional Unicode characters can be used to swap segments of text in a file. This can cause code to appear one way and be interpreted or compiled another way.

So we're seeing some early indications that the industry is treating this "Trojan Source" problem with the attention that it requires.

## SpinRite

I'm still dealing with the fallout from SpinRite's 5th technology development release. We learned something of phenomenal importance last week: There are motherboard and add-in adapter BIOSes which do not correctly report the hardware interrupt their devices use.

I received so many notes from our listeners saying that they really enjoyed my discussion of tracking down that problem with the Thinkpad whose NVMe controller had died in a really weird way, and coming up with a way to recover its BitLocker encrypted drive without any recovery key, that I'm not going to shy away from a bit of technical talk here.

Several testers of releases #4 and #5 reported hangs or timeouts on their systems. On release #5 it was many fewer than for #4 since I had found and fixed the trouble I was having with Intel chipsets when configured for ATA operation. But #5 was still having some hangs. I believe that any problem I can reproduce I can fix. And since I was unable to make that happen, I purchased one of the same adapters that one of our testers was using that was having trouble. This little IO Crest adapter used a Marvell chip and offered both serial SATA and parallel old-school PATA (IDE) connections. If drives were connected to its SATA ports, no problem. SpinRite cruised right through. But attach a drive to its IDE parallel cable and BLAMO!! A hard full system hang that only the reset button or a power cycle would end.

As I stepped through the code, instruction by instruction, the moment I tried to execute an instruction that had the controller send a command to its drive to actually do something, the debugger never returned from stepping into that instruction and the entire system was locked up. The only thing that can really do that is a hardware interrupt.

Hardware interrupts are such an incredibly powerful innovation in computing that it's difficult to imagine life without them. The UNIVAC 1103, from 1953, is generally credited with the first use of hardware interrupts. Since that was also near the birth of computing, it's clear that the early pioneers of computing themselves quickly realized the limitations of purely sequential instruction -by-instruction program flow. Normally, one CPU instruction follows the next with programmed jumps and loops both conditional and unconditional to facilitate complex logic flows.

A "hardware interrupt" does exactly what it's name says: Some hardware event interrupts that pre-programmed flow of instructions. When the interrupt occurs, the CPU's program counter is saved and a new program location is loaded into it. Thus, the hardware event causes the computer to instantly jump to some other location of its code. An example might be to count the ticks of a hardware clock to maintain the time of day. Once that clock tick has been counted, the hardware interrupt would be ended — we say that the interrupt has been "serviced" — by restoring the program counter to its previously saved value, which causes the CPU to resume executing code from the point of its interruption as if nothing had happened.

We old timers here will recall the early days of the PC with its ISA bus when interrupts were a real problem. There were only 15 hardware interrupts available back then, with the system's clock and keyboard always occupying the first two. But pretty much every peripheral added — com ports, printer ports, the screen, the floppy drive, hard disk drives, etc. — each needed to have their own dedicated hardware interrupt request signal — or IRQ. So juggling those was often a challenge.

Now it's 2021 and executing an instruction that would generate an interrupt was locking up the system. So exactly as with the early UNIVAC 1103 in 1953, the CPU was being yanked away from executing my code... only, in this case, the CPU was never returning. So, I tested the theory that an interrupt was causing the trouble by disabling the drive's hardware interrupts. It's possible to tell a drive not to generate an interrupt when it needs attention. And, sure enough, no hang. To be sure, I re-enabled the drive's interrupts, but disabled the entire system's interrupt servicing system. That's possible because there are times when one's code absolutely positively must not be interrupted for any reason. And again, no hang. So that told me where the trouble was.

So I dug in and discovered that the controller's BIOS was reporting that its controller interrupted on IRQ5 when it was actually interrupting on IRQ11. That mattered. I was waiting for the drive to signal its completion on IRQ5, so I had placed code there ahead of time to receive and handle that interruption. But instead, by signaling its completion on IRQ11, when I single-stepped into that instruction, control of the processor was yanked over to whatever code might have been handling IRQ11, if any, and disaster struck.

This sort of problem never occurred in any previous versions of SpinRite because all previous SpinRites have used whatever BIOS was present, and the BIOS always knew the truth itself, even if it wouldn't tell the truth to anyone else when asked.

So, having obtained absolute proof that there are motherboard firmware and add-in adapter BIOSes out in the world that do not accurately report the IRQ that's being signalled by their hardware, I needed to switch to Plan B.

As of the end of this past weekend, SpinRite no longer cares at all which hardware IRQ is signalled for completion. Since IRQ 0 and 1 are permanently connected to the clock and the keyboard, SpinRite now monitors all of the remaining 13 interrupt lines at once, in parallel. And as a result, SpinRite is now working perfectly on that controller. Once I publish the next dev release we'll see how many other similar appearing problems have also been resolved with this change. It feels like we're getting close.

# Bluetooth Fingerprinting

https://cseweb.ucsd.edu/~nibhaska/papers/sp22_paper.pdf

Seven researchers from the University of California at San Diego have completed some very important privacy-related research which will be formally presented during the upcoming "2022 IEEE Symposium on Security and Privacy."

The title of their paper provides the first hint into what they've found: "Evaluating Physical-Layer BLE Location Tracking Attacks on Mobile Devices." The key is their use of the term "physical layer" rather than "logical" or "data" or "application" layer. In other words, something about the WAY the Bluetooth radio is transmitting not WHAT the Bluetooth radio is transmitting.

Here's how they introduce their work in their paper's Abstract:

> *Mobile devices increasingly function as wireless tracking beacons. Using the Bluetooth Low Energy (BLE) protocol, mobile devices such as smartphones and smartwatches continuously transmit beacons to inform passive listeners about device locations for applications such as digital contact tracing for COVID-19, and even finding lost devices. These applications use cryptographic anonymity that limit an adversary's ability to use these beacons to stalk a user.*
>
> ***However, attackers can bypass these defenses by fingerprinting the unique physical-layer imperfections in the transmissions of specific devices.***
>
> *We empirically demonstrate that there are several key challenges that can limit an attacker's ability to find a stable physical layer identifier to uniquely identify mobile devices using BLE, including variations in the hardware design of BLE chipsets, transmission power levels, differences in thermal conditions, and limitations of inexpensive radios that can be widely deployed to capture raw physical-layer signals.*
>
> *We evaluated how much each of these factors limits accurate fingerprinting in a large-scale field study of hundreds of uncontrolled BLE devices, revealing that physical-layer identification is a viable, although sometimes unreliable, way for an attacker to track mobile devices.*

So what we're talking about here is exploring the feasibility of fingerprinting individual Bluetooth radios located in our consumer devices, using subtle differences in individual device RF emissions.

It turns out that this is far from the first time researchers have considered and looked into the possibility of doing this. Here are some titles of papers submitted and presented during various security, communications and engineering conferences:

2006: "Detecting Rogue Devices in Bluetooth Networks using Radio Frequency Fingerprinting." The International Conference on Communications and Computer Networks.

2007: "Implications of Radio Fingerprinting on the Security of Sensor Networks." The Third International Conference on Security and Privacy in Communications Networks and the

Workshops.

2008: "Wireless Device Identification with Radiometric Signatures."
Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, MobiCom '08.

2008: "Using Spectral Fingerprints to Improve Wireless Network Security."
IEEE Global Telecommunications Conference.

2008: "Passive Steady State RF Fingerprinting: A Cognitive Technique for Scalable Deployment of Co-Channel Femtocell Underlays."
In 2008 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks.008.

2009: "Physical-Layer Identification of RFID Devices."
In Proceedings of the 18th Conference on USENIX Security Symposium.

2011: "Identifying Wireless Users via Transmitter Imperfections."
IEEE Journal on Selected Areas in Communications.

2020: "Aircraft Fingerprinting Using Deep Learning."
28th European Signal Processing Conference.

So, the idea of fingerprinting a radio frequency transmitter by closely and carefully characterizing the details of its transmission — not the data it's transmitting but the precise way it's transmitting — is a well established question... and a potential problem for our privacy.

Since the time that researchers began looking into all of this, we've all taken to carrying individual radios in our pockets, each of which is deliberately and constantly broadcasting RF beacon signals. The designers of these technologies have gone out of their way with all sorts of fancy state-of-the-art crypto to cleverly and deeply anonymize the **data** that's being transmitted. But they've completely missed and skipped over the truth that inter-device differences may be sufficiently distinctive to render those devices individually identifiable.

That's the question that this UC San Diego team set out to answer.

*The mobile devices we carry every day, such as smartphones and smartwatches, increasingly function as wireless tracking beacons. These devices continuously transmit short-range wireless messages using the Bluetooth Low Energy (BLE) protocol. These beacons are used to indicate proximity to any passive receiver within range. Popular examples of such beacons include the COVID-19 electronic contact tracing provided on Apple and Google Smartphones as well as Apple's intrinsic Continuity protocol, used for automated device hand-off and other proximity features.*

*However, by their nature, BLE wireless tracking beacons have the potential to introduce significant privacy risks. For example, an adversary might stalk a user by placing BLE receivers near locations they might visit and then record the presence of the user's beacons. To address these issues, common BLE proximity applications cryptographically anonymize and periodically rotate the identity of a mobile device in their beacons. For instance, BLE devices periodically*

*re-encrypt their MAC address, while still allowing trusted devices to determine if these addresses match the device's true MAC address. Similarly, COVID-19 contact tracing applications regularly rotate identifiers to ensure that receivers cannot link beacons from the same device over time.*

*While these mechanisms can foreclose the use of beacon content as a stable identifier, attackers can bypass these countermeasures by fingerprinting the device at a lower layer. Specifically, prior work has demonstrated that wireless transmitters have imperfections introduced in manufacturing that produce a unique physical-layer fingerprint for that device (for example, Carrier Frequency Offset and various carrier modulation defects). Physical-layer fingerprints can reliably differentiate many kinds of wireless chipsets including a recent attempt to distinguish among 10,000 WiFi chipsets.*

[That last bit of research published last year was titled "Deep Learning for RF Fingerprinting: A Massive Experimental Study." IEEE Internet of Things Magazine.)

*To the best of our knowledge, no prior work has evaluated the practicality of such physical-layer identification attacks in a real-world environment. Indeed, prior to BLE tracking beacons, no mobile device wireless protocol transmitted frequently enough—especially when idle—to make such an attack feasible. Additionally, there is no existing BLE fingerprinting tool that can measure the physical-layer imperfections in BLE transmissions accurately. Prior techniques for fingerprinting either provide low precision fingerprints because they use short duration transient signal features, or provide high precision fingerprints but require long duration signal features which exist only in protocols like WiFi but not in BLE. Our first contribution is a tool that uses a novel method to recover these imperfections by iteratively adding imperfections to a re-encoded clean copy of a received packet, until they match the imperfections of the received packet over the air.*

[So, in other words, in order to determine the feasibility of actually doing this, they first needed to develop the best technology possible to do so. And they did.]

*Our next contribution is an evaluation of how practical it is for an attacker to track BLE-beaconing devices using their RF fingerprint. Namely, using lab-bench experiments, we identify four primary challenges to identifying BLE devices in the field: (1) BLE devices have a variety of chipsets that have different hardware implementations, (2) applications can configure the BLE transmit power level, resulting in some devices having lower SNR BLE transmissions, (3) the temperature range that mobile devices encounter in the field can introduce significant changes to physical-layer impairments, and (4) the low-cost receivers that an attacker can use in the wild for RF fingerprinting are not significantly less accurate than the tools used in prior studies.*

*Our final contribution is a set of field experiments to evaluate how significantly these challenges diminish an attacker's ability to identify mobile devices in the field. We leverage the fact that BLE tracking beacons are already used on many mobile devices. We perform an uncontrolled field study where we evaluate the feasibility of tracking BLE devices when they are operating in public spaces where there are hundreds of other nearby devices. To the best of our knowledge, our work is the first to evaluate the feasibility of an RF fingerprinting attack in real-world scenarios.*

*We show that even when there are hundreds of devices we encountered in the field, it is still*

*feasible to track a specific mobile device by its physical-layer fingerprint.*

*However, we also observe that certain devices have similar fingerprints to others, and temperature variations can change a device's metrics. Both of these issues can lead to significant misidentification rates.*

*In summary, we find that physical layer tracking of BLE devices is indeed feasible, but it is only reliable under limited conditions, and for specific devices with extremely unique fingerprints, and when the target device has a relatively stable temperature.*

The wonderfully detailed paper goes on for 15 pages and for anyone interested it provides all the detail anyone could want. All of their work and datasets is also posted on GitHub.

I'm tempted to conclude that there's nothing for us to worry about overall. And for the most part that's the reasonable conclusion. But one part of their paper stuck out and caught my attention. They conducted a case study of tracking a specific person. I'll read what they wrote then clarify the tricky bits:

*Case Study 2: Tracking a person:*
*We conducted an end-to-end tracking attack executed on a controlled target (a volunteer who uses an iPhone). The attacker first carries their SDR* [software defined radio] *sniffer close to the target device to obtain the device's physical-layer fingerprint.*

*Simultaneously, the attacker scans for nearby BLE devices using a commonly available BLE scanner phone app, and they record the MAC address of the BLE device with the highest observed signal strength, which is the nearest device (i.e., the target's phone). When later post-processing all of the data and signals collected, they use the target's MAC address to pick out the target device's packets from the raw sniffer capture. They feed these packets into the BLE tracking toolkit to train its classifier with the target device's fingerprint.*

*After creating the fingerprint, the attacker tracks their target by placing an SDR and laptop close to their target's home. The attacker can determine when the target is home by observing when the classifier running on the laptop indicates the packets received by the SDR match the target device's fingerprint. The attacker tracks their target for one hour, during which the target walks inside and outside the house 2 times. Figure 18* [below] *shows the number of unique MAC addresses observed every ten seconds during this hour. There are approximately 30 other devices nearby that could be confused with the target.*

*The blue bar shown in Figure 19* [below] *shows the ground truth of when the person was inside the house during this hour. The attacker's identification toolkit runs once every 10 seconds, and the red bar shows the time durations during which the tracking toolkit thinks the person was present. The bars perfectly match except for immediately prior to minute 10, where the toolkit falsely detects the presence of the target for 50 seconds, even though it had not yet actually returned.*

I've included the two figures referred to in their research below. That Figure 18 shows the number of unique MAC addresses observed during the hour long test and Figure 19 shows their detection chart which, except for that brief false-positive mis-detection, is perfect:

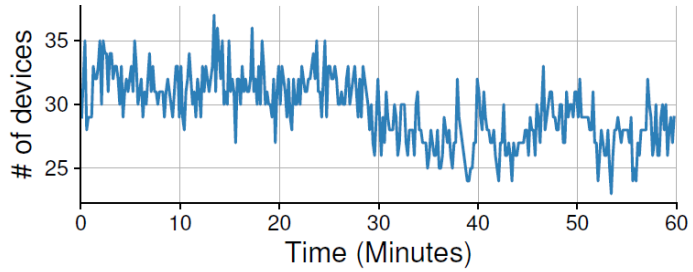



Fig. 18: Number of unique MAC addresses observed over time while tracking the target.

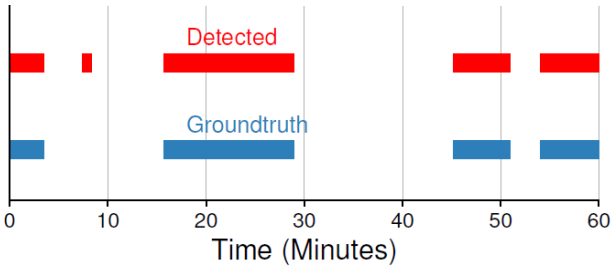



Fig. 19: The blue bar represents the time that the target was present, the red bar represents the time that our tracking toolkit detected the presence of the target.

This was significant because it suggests that, while the natural variations in RF signal generation are not generally sufficient to identify arbitrary individuals within a large population, it may well be feasible to train an inexpensive RF classifier to recognize a specific Bluetooth radio with sufficiently high accuracy that the same radio will later be recognized with a high probability of success. You can image that law enforcement and counter-terrorism agencies might find ample application for such passive device recognition.

So what are our takeaways from this work?

This has usefully highlighted a weak, but potentially significant flaw, in the assumption that the only thing we need to be concerned about for protecting our anonymity and privacy is the digital data our radios transmit. It's now clear that subtle differences in the analog way they transmit can be significant. And it feels as though this is the sort of thing that Apple and Qualcomm might find to be of sufficient concern that they might consider adding some deliberate noise into their radio modulation channels to thoroughly thwart this form of de-anonymization attack.