Security Now! #991 - 09-10-24

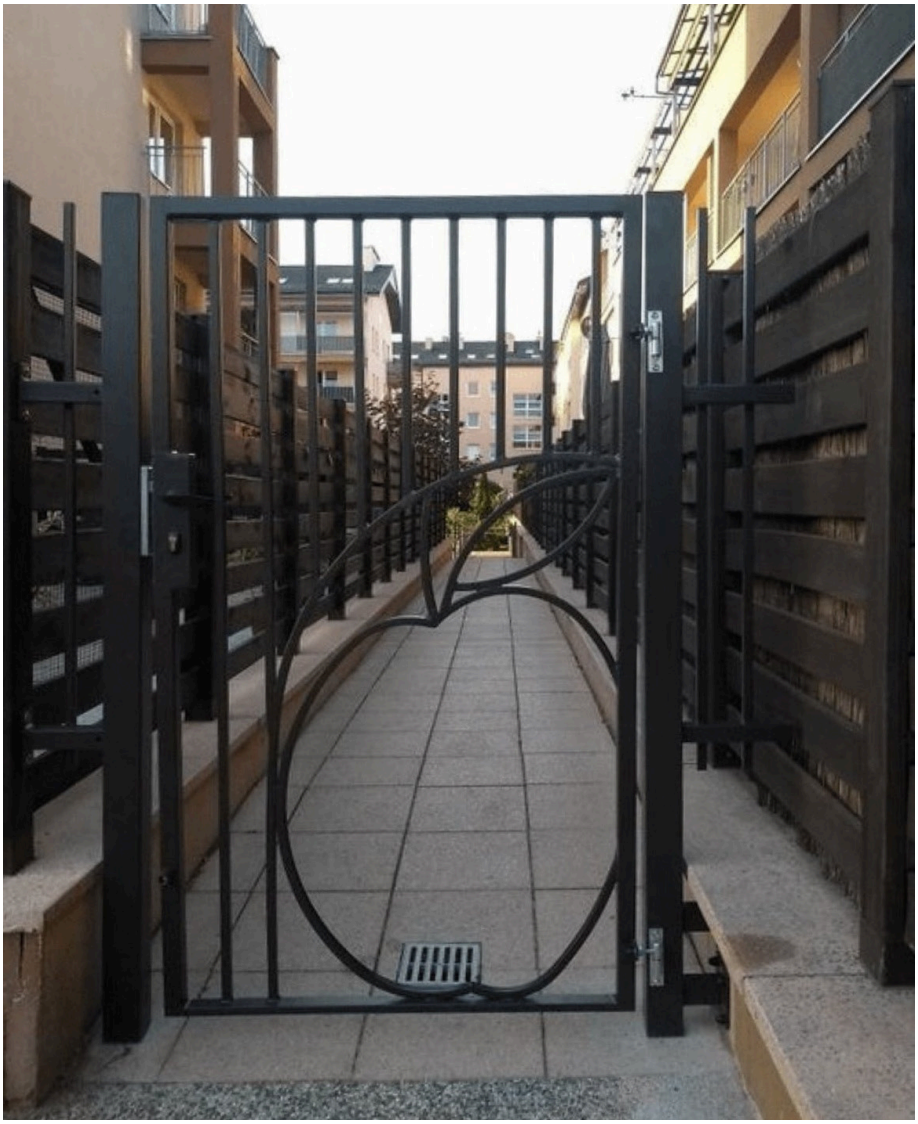
RAMBO

This week on Security Now!

Microsoft's "Recall" uninstallability is a bug. Yubikeys can be cloned. How worried should you be? When was that smoke detector installed? We share and discuss lots of interesting listener feedback: Is whatsApp more secure than Telegram? Does Telegram's lack of security really matter? Elevators in Paris have problems, too. There's a 4th credit bureau to be frozen, too. Can high pitched sound keep dogs from barking? A reminder of a terrific UNIX 2038 countdown clock. A new Bobiverse Sci-Fi book & new Peter Hamilton novel. Why does SpinRite show user data flashing past? And... TEMPEST is alive and well in the form of the latest RAMBO attack.

The very definition of "Form over Function"

Note the extra bars on the exterior of the gate to prevent anyone from sneaking around it! Right.

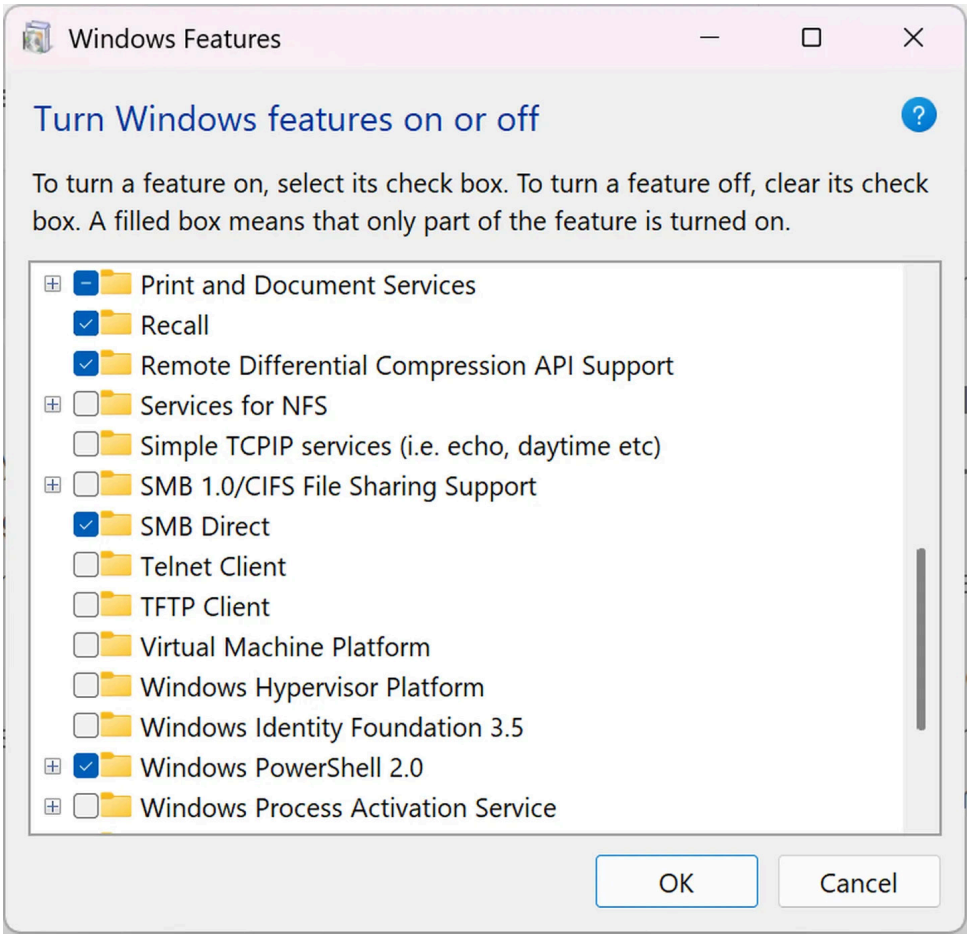


Security News

Offer to uninstall Recall was a bug, not a feature

The Verge carried some news that really makes you wonder what's going on at Microsoft. Their headline read: "Microsoft says its Recall uninstall option in Windows 11 is just a bug" – in other words, don't get your hopes up that we're going to allow our illustrious forthcoming "Recall" feature to be removed from Windows – that was a bug, not a feature. The Verge writes:

> *While the latest update to Windows 11 makes it look like the upcoming Recall feature can be easily removed by users, Microsoft tells us it's just a bug and a fix is coming. Deskmodder spotted the change last week in the latest 24H2 version of Windows 11, with KB5041865 seemingly delivering the ability to **uninstall** Recall from the Windows Features section.*



I have a picture of the "Windows Features" dialog in the show notes and there it is, right underneath "Print and Document Services" and above "Remote Differeential Compression API Support" – "Recall". It's currently checked showing that it's installed and offering to be unchecked and thus removed. But no, that's a bug!

In a statement to The Verge, Windows senior product manager Brandon LeBlanc said: *"We are aware of an issue where Recall is incorrectly listed as an option under the 'Turn Windows features on or off' dialog in Control Panel. This will be **fixed** in an upcoming update."*

The Verge goes on to tell us much of what we already know, which is to say why many of us would wish that checkbox would remain. But The Verge also adds a bit of news. They wrote:

> *The controversial Recall AI feature, which creates screenshots of mostly everything you see or do on a computer, was originally supposed to debut with Copilot Plus PCs in June. Microsoft was forced to delay the feature after security researchers raised concerns. Microsoft says it remains on track to preview Recall with Windows Insiders on Copilot Plus PCs in October, after the company has had more time to make major changes to Recall.*
>
> *Security researchers initially found that the Recall database that stores the snapshots of your computer every few seconds wasn't encrypted, and malware could have potentially accessed the Recall feature. Microsoft is now making the AI-powered feature an opt-in experience instead of on by default, encrypting the database, and authenticating through Windows Hello.*
>
> *We did ask Microsoft whether it will allow Windows users to fully uninstall Recall, as this appearance in the Windows features list suggests, but the company only confirmed this was just "incorrectly listed" for now. It's possible that Microsoft may **need** to add a Recall uninstall option to EU copies of Windows 11 to comply with the European Commission's Digital Markets Act. Microsoft has already had to add an uninstall option for Edge in European Economic Area (EEA) countries, alongside the ability to remove the Bing-powered web search in the Start menu.*

And really, when you think about it, what does it mean that Windows has a feature that presents a clear and present privacy and security danger to all of its users, which Microsoft knows full well many of its users feel extremely uncomfortable about. And where it's obvious that that feature could be readily removed from Windows... but Microsoft refuses to allow their users to do so.

One thing that means for certain, is that GRC's forthcoming freeware which will totally neuter and remove Recall, promises to be quite popular.

YubiKeys can be cloned

I probably received as many pointers to the recent Yubikey exploit stories from our listeners as I did to the news of the new RAMBO attack. This is, of course, due to the fact that Yubico largely credits me, thanks to the listeners of this podcast, with discovering them at an RSA conference where I met Yubico's primary mover, shaker and co-founder, Stina Ehrensvärd, and then with putting them on the map and getting them going. It's clear that this would certainly have happened for Yubico sooner or later (and probably sooner). It was just fortune that I happened to be someone with a microphone who recognized the cleverness of what they had created.

ArsTechnica's headline about the recent discovery was: *"YubiKeys are vulnerable to cloning attacks thanks to newly discovered side channel"* and their sub-head reads: *"Sophisticated attack breaks security assurances of the most popular FIDO key."*

The researchers at NinjaLab, who performed the research and earlier informed Yubico of their findings, wrote:

In the present work, NinjaLab unveils a new side-channel vulnerability in the ECDSA [Elliptic Curve Digital Signature Algorithm] implementation of Infineon 9 on any security microcontroller family of the manufacturer. This vulnerability lies in the ECDSA ephemeral key (or nonce) modular inversion, and, more precisely, in the Infineon implementation of the Extended Euclidean Algorithm (EEA for short). To our knowledge, this is the first time an implementation of the EEA is shown to be vulnerable to side-channel analysis (contrarily to the EEA binary version). The exploitation of this vulnerability is demonstrated through realistic experiments and we show that an adversary only needs to have access to the device for a few minutes. The offline phase took us about 24 hours; with more engineering work in the attack development, it would take less than one hour.

After a long phase of understanding Infineon implementation through side-channel analysis on a Feitian 10 open JavaCard smartcard, the attack is tested on a YubiKey 5Ci, a FIDO hardware token from Yubico. All YubiKey 5 Series (before the firmware update 5.7 11 of May 6th, 2024) are affected by the attack. In fact all products relying on the ECDSA of Infineon cryptographic library running on an Infineon security microcontroller are affected by the attack. We estimate that the vulnerability exists for more than 14 years in Infineon top secure chips. These chips and the vulnerable part of the cryptographic library went through about 80 CC certification evaluations of level AVA VAN 4 (for TPMs) or AVA VAN 5 (for the others) from 2010 to 2024 (and a bit less than 30 certificate maintenances).

In his reporting of this, Dan Goodin, writing for ArsTechnica notes:

The attacks require about $11,000 worth of equipment and a sophisticated understanding of electrical and cryptographic engineering. The difficulty of the attack means it would likely be carried out only by nation-states or other entities with comparable resources and then only in highly targeted scenarios. The likelihood of such an attack being used widely in the wild is extremely low. Roche said that two-factor-authentication and one-time password functionalities aren't affected: because they don't use the vulnerable part of the library.

Tuesday's report from NinjaLab outlines the full flow of the cloning attack as:

- *The adversary steals the login and password of a victim’s application account protected with FIDO (e.g., via a phishing attack).*
- *The adversary gets physical access to the victim’s device during a limited time frame without the victim noticing.*
- *Thanks to the stolen victim’s login and password (for a given application account), the adversary sends the authentication request to the device as many times as is necessary while performing side-channel measurements.*
- *The adversary quietly gives back the FIDO device to the victim.*
- *The adversary performs a side-channel attack over the measurements and succeeds in extracting the ECDSA private key linked to the victim’s application account.*
- *The adversary can sign in to the victim’s application account without the FIDO device and without the victim noticing. In other words, the adversary created a clone of the FIDO device for the victim’s application account. This clone will give access to the application account as long as the legitimate user does not revoke its authentication credentials.*

The list, however, omits a key step, which is tearing down the YubiKey and exposing the logic board housed inside. This likely would be done by using a hot air gun and a scalpel to remove the plastic key casing and expose the part of the logic board that acts as a secure element

storing the cryptographic secrets. From there, the attacker would connect the chip to hardware and software that take measurements as the key is being used to authenticate an existing account. Once the measurement-taking is finished, the attacker would seal the chip in a new casing and return it to the victim.

And to put this into context, Dan adds...

The attack and underlying vulnerability that makes it possible are almost entirely the same as that allowed NinjaLab to clone Google Titan keys in 2021. That attack required physical access to the token for about 10 hours.

The attacks violate a fundamental guarantee of FIDO-compliant keys, which is that the secret cryptographic material they store can't be read or copied by any other device. This assurance is crucial because FIDO keys are used in various security-critical environments, such as those in the military and corporate networks.

That said, FIDO-compliant authentication is among the most robust forms of authentication, one that's not susceptible to credential phishing or adversary-in-the-middle attacks. As long as the key stays out of the hands of a highly skilled and well-equipped attacker, it remains among the strongest forms of authentication. It's also worth noting that cloning the token is only one of two major steps required to gain unauthorized access to an account or device. An attacker also must obtain the user password used for the first factor of authentication. These requirements mean that physical keys remain among the most secure authentication methods.

To uncover the side channel, the researchers reverse-engineered the Infineon cryptographic library, a heavily fortified collection of code that the manufacturer takes great pains to keep confidential. The detailed description of the library is likely to be of intense interest to cryptography researchers analyzing how it works in other security devices.

So what we have here is Yubico in the spotlight only because it's by far the most successful and well known user of high-security token hardware that, despite many previous reviews and extensive analysis by the industry, was finally found to have an extremely subtle flaw that could be used to extract its secrets – and even and only then, through the use of quite high-end expensive engineering equipment, including the need to physically compromise and crack open the key.

And even then, the attacker would still need knowledge that only the key's legitimate owner and user probably possesses.

Infineon has fixed their problem with a firmware update but in the interest of security, Infineon's firmware is not field upgradeable. So Yubico has obtained the improved hardware from Infineon and is now offering keys that have this fixed. Whether or not anyone should or would bother to update is up to them. But this attack seems so far-fetched, and is so far out of the realm of ever happening to anyone – and, after all, we're just using the keys to contain additional factors of logon credentials – that I can't imagine that this is worth another thought.

Miscellany

Our picture of the week, podcast before last, was that signage which was intended to have its blank field proudly filled-in with the date since there had last been any sort of accident on the job. But instead, it cited that they'd had no accidents since someone (whom everyone presumably knew) had left the job. I recalled that we'd had a similar non-sequitur once for our picture of the week in the form of a close-up photo of a smoke alarm that also had a blank space where its installer was expected to fill in a date... but during the podcast I was unable to recall what had been written there instead. One of our listeners whose online moniker is "Mr Nobody 2" was kind enough to remind me:

The smoke alarm had a field where its installation date meant to be filled-in by its installer. So it said: "Installed On:" followed by a blank space. In this case, the person filled the information in so that it read: "Installed On: The Ceiling."

This week there was not a huge amount of news. And I got caught up in the terrific listener feedback that I'd been receiving. Many years ago we used to deliberately alternate between security news and listener feedback episodes. We've dropped that approach in favor of always doing some of both, but this week we're going to spend more time over on the feedback side.

I should note how pleased I am with the way GRC's email system has worked out. The nature of the feedback by email is different from Twitter and having it in my email client makes it significantly easier to manage. So I'll remind everyone that in order to send feedback to me at the email address "securitynow@grc.com" you need to register your sending email address with GRC. You do **not** need to subscribe to any of the three mailing lists that you'll find there. Just being registered allows my system to prevent all incoming spam. But I'm also hearing from many of our listeners who really appreciate receiving the weekly show title, summary, picture of the week and show notes link by mail every Tuesday morning before the podcast. So subscribing to the Security Now! List will automatically make that happen for you.

Okay...

Closing the Loop

Angus MacKinnon

Steve: I see on WhatsApp all the time your messages are encrypted. Is Whatsapp secure? I thought WhatsApp had Signal embedded.

Last last week's podcast was all about the fact that Telegram – which claims security and boasts of its reputation for security – was not truly offering end-to-end messaging encryption, with the single exception that two – and exactly and only two – parties who were both online at the same time could deliberately enable point-to-point encryption for their conversation. So I'm sure that Angus just wanted some assurance that WhatsApp's similar claim of encrypted messaging is actually legitimate. And as he noted, since WhatsApp is based upon the open Signal protocol, all messaging is always fully encrypted, even in multi-party groups. In fact, there's no way to use Signal in any unencrypted mode.

Andy Pastuszak shared some useful points which he feels favors Telegram:

Steve, I'm a user of Telegram as well as Signal. The definition of anything less than end-to-end encryption as not being true encryption would make a LOT of services not encrypted, even outside the messaging space.

There are almost no cloud providers that offer true end-to-end encryption. Dropbox, OneDrive and Google Drive don't. Online calendar, todo lists and note taking apps don't really either. And the ones I find that do, charge A LOT for the privilege. Some of the end-to-end encrypted note-taking apps I looked at charge well over $100/year for their basic plan.

Telegram is obviously NOT e2e encrypted. But it is encrypted in transit and encrypted at rest. For the things I use Telegram for, all I really need is encryption in transit. If I really need end-to-end encryption, then I use Signal.

The other nice thing about Telegram is how group chats work. How many times have you been part of a group SMS text, and ask to be removed from it. And that works great until someone responds to an old message that you're still included on, and then all of the sudden, you're part of the conversation again. With Telegram, you leave a group chat/channel, you're gone till you rejoin.

And Telegram fully support Siri and CarPlay. I can easily say "Hey Siri, send a Telegram message to Joe" while driving and it will happily do that. Signal does not have Siri or CarPlay support yet. So if you want something better than SMS, with Siri and CarPlay/Android Auto support and you're aware of the encryption limitations, Telegram is an excellent choice.

I agree with Andy that true end-to-end encryption is rarely needed or necessary. I use iMessage among my iPhone-using friends. As we know, its encryption is what Matthew Green described as modern state-of-the-art. But the messages are about what time we're meeting for dinner or whether they saw some random piece of news. Hardly anything that would ever be of interest to anyone else.

Andy is obviously a sophisticated user who understands exactly what's going on; so there's nothing to disagree with him about. One of the points of his sophistication is that he knows that when he truly needs end-to-end encryption it's time to switch to Signal for that. But a big part of what Matthew Green wanted to convey – although it was read by people like us, so it didn't come as a huge surprise – was that the **typical** Telegram user was extremely unlikely to have any such sophistication and thus appreciation of the distinction. So Matthew told us that he was growing increasingly annoyed as the years rolled by with Telegram not making any significant improvements to the security of their messaging technology while essentially riding on the coat-tails of all of the other fully end-to-end encrypted messaging platforms, all the while claiming privacy parity while choosing to not actually offer it.

John Hickin

Steve, we rented an apartment in Paris where a sign was present in the elevator (but in French of course). It was put up by owners who were annoyed when renters (AIRBNB) forgot to close the outer door after leaving the elevator, thus rendering it stuck in place so nobody on any other floor could recall and use it. Cheers, John.

I enjoyed John's note which related, of course, to last week's picture of the week which suggested that if the elevator didn't "go" its occupants might try jumping up and down a bit, presumably to let it know they were present – although one would imagine that pressing a floor button would serve that purpose. Apparently, in Paris today, they're still using those quaint elevators where its user first closes an outer door on the floor, remains on the floor to close off the elevator shaft and the elevator is only responsible for closing and opening its own carriage door. So, as John notes, if people leaving an elevator leave the outer floor door open with the elevator unable to close it, the elevator is unable to budge without leaving the elevator shaft open when the carriage has left.

Craig Taylor

Hi Steve, Long time listener. I wanted to provide you with some additional information on the article you cite for Freezing Credit in the NDP breach. The article you reference for Freezing credit only mentions 3 of the 4 major credit bureaus at which you need to freeze your credit. Innovis is missing from that article. Our article at CyberHoot has a collection of many of the primary and secondary credit bureau's. https://cyberhoot.com/cybrary/identity-theft/
Great coverage and thanks for doing what you do!

Craig is a co-founder of CyberHoot and the page he linked to does, indeed, provide more comprehensive coverage of the various credit bureaus with links to each bureau's individual credit freeze resources page. The GRC shortcut "npd" for checking the NPD breach database received the largest number of referring clicks **ever**, and GRC's "credit" link to the investopedia page is next in line, just behind it, in 2nd place runner up position. So I know this topic is, not surprisingly, of significant interest to this podcast's listeners. Since I want to make Craig's more comprehensive listing of credit bureau credit freeze links readily accessible, I've created another GRC shortcut, this one is "freeze" which points to Craig's excellent page about identity theft. So the link is: grc.sc/freeze.

Since I had only previously frozen my credit at the big 3 – TransUnion, Experian and the infamous Equifax – I immediately used the new link to Craig's page to find the link to Innovis, went there and froze my credit. To Innovis' credit (pardon the pun), it was the easiest of any of the freezing experiences. No need to create any account. You just fill out an online form (containing all of the data that's already public in the breach) and your credit is immediately frozen against anyone's inquiry. Innovis then sends, by postal mail, a credit freeze confirmation letter which contains a 10-digit PIN. That PIN can subsequently be used to manage your freeze status at Innovis. It was so quick and easy that I cannot imagine why anyone who cares about this would not do it. So, again, the GRC shortcut to get to Craig's page at CyberHoot is grc.sc/freeze.

And I should mention that Craig's quite comprehensive page mentions an additional five lesser bureaus which also offer credit freezing. I didn't bother with them, but if you want to be fully covered you may wish to. And Craig, thanks very much for bringing this additional major bureau to our listener's attention. Much appreciated!

Adam Tyler

Hi Steve, I was curious if you or a listener have found a commercial version of the portable dog killer device? I'm not really looking for a laser gun, but something that could sit on the fence line to deter a barking dog. Ideally automatically activated and a battery design that made sense. Lithium ion with a little solar panel would be sweet. Anyway, love the podcast, glad you are going past 999. I also only had an X/Twitter account to DM you and am very happy to see you've moved over to email. Regards, Adam Tyler

Adam is, of course, referring to one of this podcast's favorite past episodes which we've re-aired a number of times through the years because it tells a fun story which ends with a moral of the surprising benefits that can arise from being active rather than passive. I first shared that youthful adventure on the occasion of the 50 anniversary of the laser. The device I designed and built when I was in high school was not a laser, though the beam of high intensity directed sound energy it produced was likely coherent.

Twelve years ago, back in 2012 when this podcast was only 7 years old, I recreated that device after so many of our listeners commented that their neighbors' barking dogs were ruining their lives. Since I didn't have the web forum technology running that I have today, I created a Google group called "Portable Sound Blaster" for public discussion of this, and I published the final electronic design of the device on a page at GRC, naming the project "The Quiet Canine." If you're curious, you can find it under GRC's website menu under "Other" and "The Quiet Canine". https://www.grc.com/tqc/TheQuietCanine.htm

On that page I wrote: *"The good news is that we arrived at an extremely simple, inexpensive, and easy-to-build design for a small, lightweight and painfully loud handheld sound emitter."* And then the page shows the design. But then, under the caption "The Bad News" I wrote:

"Many of these final "TQC v2.2.2" devices were assembled and tested by those following and participating in the Portable Sound Blaster group at Google. The devices were invariably incredibly loud and high pitched. While their dads were assembling and testing the devices downstairs in the garage, their upstairs teenagers were complaining about the piercing sound penetrating their heads. And, of course, dogs were at least as well able to hear it, and at much greater distance.

But in no event was this able to function as any sort of barking deterrent. Dogs heard it, and at any distance, they didn't care. We soon came to appreciate that my own original "point blank" blasting of the original "Portable Dog Killer" (as I named my first device when I was in high school) was required for the device's effectiveness. No dog next door, let alone down the block, will care about a high pitched sound. It needs to be blasted directly into the dog's face at a very short distance.

This means that while this device would not be useful for silencing dogs at a distance, it would likely be extremely useful and effective as a personal defense device for walkers, postal workers on foot and joggers who are harassed and threatened by overly aggressive canines on the loose. Although we cannot and do not offer any specific guarantees, it is difficult to see how any attacking dog would not be stopped in its tracks by a close blast of incredibly loud and high pitched sound."

So the bottom line is, my particular use-case turned out to be unique. I designed and used the first device back in the early 1970's specifically to train an incredibly aggressive dog not to jump on the fence bordering the sidewalk which was terrifying passers by and causing them to fall off the sidewalk into the street. And as we know, it also had the side effect of altering the flight path of seagulls at a distance. But what it has never succeeded in doing, to my knowledge, is silencing any barking dogs; in fact, it probably encourages their barking.

All that said, thanks to this history, I occasionally receive email from listeners who stumble upon commercial solutions claiming to solve this problem. I never pay them any attention, so I have not gathered any links or referrals since I'm skeptical that any of them do anything to solve the problem. So, sharing that story was a lot of fun, as was recreating a super-loud high frequency sound generator. Mostly though, I learned just how much upset barking dogs are causing today. I wish I could offer a solution.

John,

> *Hey Steve, I stumbled across this very cool looking hexadecimal clock face with ticking hands showing the time in the venerable Unix time, and thought you, Leo and the rest of the listeners would love to see too. Check it out at* https://retr0.id/stuff/2038/
> *All the best to 999 and well beyond, John*

So, we have previously encountered this wonderful version of the UNIX clock. Thinking that I would probably have created a GRC shortcut for it previously, and I found it created almost exactly two years ago on September 18th, 2023. And the shortcut itself is, not surprisingly, "2038" so: grc.sc/2038.

UNIX time is represented by a 32-bit signed integer which has been incrementing once per second since midnight of January 1st, 1970. In what's known as signed two's complement format, the most significant bit of a number's binary representation is reserved for the number's positive or negative sign, with the bit set to '1' for negative numbers. This works out naturally when doing 2's complement binary math, which is the system used by all contemporary computers. For example, subtracting 10 from 5 should produce negative 5, and that's what happens if negative values have their high bit set. However, UNIX time could and arguably should have been defined as an unsigned 32-bit integer since it was meant to be used for timekeeping into the future, not the past. But as it is, the result of UNIX time being a signed value means that negative values represent times before 1970, extending back to 1901... which is not highly useful for things like timestamping database entries.

The good news is that all modern UNIX-like systems, and even the UNIX's themselves, have long ago switched to 64-bit time representations. But as we always see, there are surprising corners of technology that are slow to update. So it's entirely foreseeable that there will be some breakage somewhere when we finally get to 2038, 14 years from now.

This specific clock is very cool and very nerdy, thus very appealing, since those 32 bits are broken into four, 8-bit bytes with each of the 4 bytes determining the position of each of the clock's four hands. Since an 8-bit byte can have any one of 256 values, the clock has 256 "ticks" around its face. And since trouble begins once the high-byte, represented by the red hand,

reaches its half-way point, straight down, this graphic makes it very clear that we're well on our way toward the UNIX apocalypse.

I would dearly love to still be doing this podcast 14 years from now and to be able to cover and discuss the events of the end of 32-bit UNIX time. I would not be surprised if some things break.

Norbert,

> *Bobiverse book #5 came out on Sept 5: "Not Till We Are Lost" Just want to let you know. Thanks for the great postcase ! Norbert*

Thanks Norbert! I know that the Bobiverse series has been a huge hit with our listeners, so I wanted to share the news of book #5's availability with everyone.

Exodus: The Archimedes Engine

The Bobiverse books were pretty easy to breeze through. But for anyone who's interested in really sinking their teeth into something that promises to be far more substantial, our listener Simon Zerafa sent me a note that one of this podcast's favorite Sci-Fi authors, none other than the great Peter F. Hamilton, is releasing his next novel next week. That's the good news. What may be bad news depending upon your need for more immediate closure, is that this is book number one of a two-part novel series. In the past, as with, for example, Pandora's Star which left us hanging quite a while for story's conclusion in "Judas Unchained", and later it was the same with Peter's Dreaming Void series, Peter is famous for laying down a lot – and I mean really a lot – of foundation in his novels, so that things are finally really starting to move just as the first novel is ending. That may not bother everyone, but it bugs the crap out of me. So I'm sure I'll be waiting for the publication of the series' conclusion because to purchase both books so that I'm able to read them back to back.

The first book's title is: "Exodus: The Archimedes Engine" and the synopsis, probably taken from the back cover of the hardback, so just to give its reader a sense for what's to come while not being a spoiler, reads:

> *Forty thousand years ago, humanity fled a dying Earth. Traveling in massive arkships, these brave pioneers spread out across the galaxy to find a new home. After traveling thousands of light-years, one fleet of arkships arrived at Centauri, a dense cluster of stars with a vast array of potentially habitable planets. The survivors of Earth signaled to the remaining arkships that humanity had finally found its new home among the stars.*
>
> *Thousands of years later, the Centauri Cluster has flourished. The original settlers have evolved into advanced beings known as Celestials and divided themselves into powerful Dominions. One of the most influential is that of the Crown Celestials, an alliance of five great houses that controls vast areas of Centauri. As arkships continue to arrive, the remaining humans and their descendants must fight for survival against overwhelming odds or be forced into serving the Crown Dominion.*

Okay. So it sounds as though this Crown Dominion is old and corrupt.

Among those yearning for a better life is Finn, for whom Earth is not a memory but merely a footnote from humanity's ancient history. Born on one of the Crown Dominion worlds, Finn has known nothing but the repressive rule of the Celestials, though he dreams of the possibility of boundless space beyond his home.

When another arkship from Earth, previously thought lost, unexpectedly arrives, Finn sees his chance to embrace a greater destiny and become a Traveler—one of a group of brave heroes dedicated to ensuring humanity's future by journeying into the vast unknown of distant space.

Okay. So at this point this is not any sort of recommendation because I haven't yet read the book. I am certain I'll read both book once they become available. But if anyone listening does decide to jump on the first book knowing that they may be left with a classic Hamilton cliffhanger, please DO send your review to me at "securitynow@grc.com" and I'll share what you think without any spoilers.

Hadrian,

Hi Steve, Long time reader, then listener, then viewer. I recently bought SpinRite. Not yet needed for recovery, but I now have a burning question: am I the only one who looks at the raw data display and then suddenly says "Hey! I know which file that was!" ??

I got a kick out of Hadrian's note because, though no one else has ever mentioned it specifically that I can recall, I, too, will often see something I recognize flash past on SpinRite's Real Time Activities display. But SpinRite did not always show that. Back before mass storage drives were able to manage their own defective sectors, SpinRite needed to and did handle all of that itself. This meant that sectors embedded in clusters that had been found to be defective would need to be relocated then replaced by good clusters. So that region of SpinRite's Real Time Activities UI page once tracked all of those changes and showed totals by count and bytes of everything that SpinRite had done. At some point, once all drives became able to handle defect relocation autonomously, although SpinRite would still induce a drive to perform the relocation, now that would happen below the level of the file system. So I was able to remove all of that logic from SpinRite. That also meant that I needed to remove all of the tracking, totalling and displaying of that work which SpinRite no longer needed to do. And that left a big empty display region in SpinRite's user interface. I decided to fill that hole with an updating snapshot of the data that was passing by, so that SpinRite's user could literally see the data that SpinRite was working on. This has become one of SpinRite's more popular user-interface features.

The final piece of feedback leads us nicely into this week's topic.
The feedback was sent by ***a U.K. listener named Laura***, who wrote:

Hi Steve, My name is Laura, from the UK. As I have a Masters degree in Cyber Security, I came across this article and hope you would be interested in talking about this both for me and everyone else.

I love the show and I'm so glad you are going past 999 as I have a standing appointment with you and Leo every Tuesday night – that no-one is allowed to interrupt. (My ex tried.) I have included the link below: https://cybersecuritynews.com/rambo-attack-air-gapped-systems/

Thank you again, Laura (P.s Leo love the new attic)

RAMBO

Many of our listeners forwarded news to me of this latest side-channel attack brought to us by none other than another clever researcher at Israel's Ben-Gurion University of the Negev. It was easy to see how much attention this latest bit of research drew, since the many links I received from our listeners – and thank you all, by the way, for sending them – you all essentially voted for this week's topic – were from widespread cyber-security related publications. Before I dug into what it was all about I was hoping that the reason for all the attention was not only because the new attack was named "RAMBO" … and I was not disappointed. So I decided that RAMBO should be this week's main discussion topic. And, also, everyone knows that I have a difficult time ignoring access to the raw research. The worst case is having to decipher something that a public relations person wrote. But in this case we have 18 pages of pure delicious research written by the researcher Mordechai Guri, which explains his new attack in full detail.

The Abstract of Mordechai's research says:

> *Air-gapped systems are physically separated from external networks, including the Internet. This isolation is achieved by keeping the air-gap computers disconnected from wired or wireless networks, preventing direct or remote communication with other devices or networks. Air-gap measures may be used in sensitive environments where security and isolation are critical to prevent private and confidential information leakage.*
>
> *In this paper, we present an attack allowing adversaries to leak information from air-gapped computers. We show that malware on a compromised computer can generate radio signals from memory buses (RAM). Using software-generated radio signals, malware can encode sensitive information such as files, images, keylogging, biometric information, and encryption keys. With software-defined radio (SDR) hardware, and a simple off-the-shelf antenna, an attacker can intercept transmitted raw radio signals from a distance. The signals can then be decoded and translated back into binary information. We discuss the design and implementation and present related work and evaluation results. This paper presents fast modification methods to leak data from air-gapped computers at 1000 bits per second. Finally, we propose countermeasures to mitigate this out-of-band air-gap threat.*

The first thing I'll note is that while 1000 bits per second won't allow you to send Windows, or even a Windows update over the air, a modern state of the art cryptographic private key is only several kilobits in length, so the keys to the kingdom could be broadcast from just such a compromised machine, over and over, every few seconds. And since it would just appear as random RF noise, no one would ever be the wiser. And unlike most malicious code whose purpose is readily revealed through inspection, any code that's being used to generate radio signals from memory buses will just be puzzling for any forensics researchers. They'd stare at it and scratch their heads and never have any idea what the heck such code was doing. They couldn't even ever be certain that it was doing anything malicious. It wouldn't appear to be doing anything at all since the designers of this code are using a far-fetched side effect of normal data processing to get their message out of the machine.

The second thing to note is that one of the consequences of today's heavy use of encryption is that we've grown to rely upon it completely. What this means, practically, is that today we're far less worried about storing our sensitive encrypted data in far more accessible places – such as in

the ubiquitous cloud. *"Who cares if it's in the cloud. It's encrypted, right?"* Sure thing. That's true, right up until the time someone figures out how to exfiltrate the comparatively tiny secret key that's protecting the otherwise far less secured data.

So my point is, thanks to the application of cryptography virtually everywhere today, we now concentrate vastly more value into a handful of bits. So whereas 1000 bits per second cannot be used to transfer a massive database, if those few thousand bits are the ***secret*** that's protecting a massive database in the cloud, then a few seconds worth of transmission is all that's needed to crack that database wide open.

Reminding us that air-gapping and air-gap exploits have a significant and deep history, Mordechai explains:

> *Enforcing an air gap in a computing or networking environment involves physically and logically isolating a system, network, or device from external networks or communication channels. This can be done by disconnecting network cables, disabling wireless interfaces, and disallowing USB connections. In addition, it must be ensured that the isolated system has no direct link to any external communication infrastructure.*
>
> *Despite air-gapped networks being considered highly secure, there have been incidents demonstrating that air-gapped networks are not immune to breaches. Stuxnet is one of the most famous air-gap malware. Discovered in 2010, Stuxnet was a highly sophisticated worm that targeted industrial control systems (ICS), particularly those used in nuclear facilities. It exploited zero-day vulnerabilities and used several methods, including infected USB drives, to jump the air gap and spread it across isolated networks.*
>
> *The Agent.BTZ worm was another type of air gap computer worm with advanced capabilities and a targeted type. It was specifically designed to spread through removable media, such as USB flash drives, and infiltrate computer networks, including those highly secure or air-gapped. According to reports, the worm affected the U.S. Department of Defense classified networks. Notably more than twenty-five reported malware in the past targeted highly secured and air-gapped networks, including USBStealer, Agent.BTZ, Stuxnet, Fanny, MiniFlame, Flame, Gauss, ProjectSauron, EZCheese, Emotional Simian, USB Thief, USBFerry, Retro, and Ramsay.*

And then Mordechai discusses his new air-gapped attack:

> *In order to exfiltrate information from an infected air-gapped computer, attackers use special communication channels known as air-gap covert channels. There are several types of covert channels studied in the past twenty years. These attacks leak data through electromagnetic emission, optical signals, acoustic noise, thermal changes , and even physical vibrations. In this paper, we show how malware can manipulate RAM to generate radio signals at clock frequencies. These signals are modified and encoded in a particular encoding allowing them to be received from a distance away.*
>
> *The attacker can encode sensitive information (keylogging, documents, images, biometric information, etc.) and exfiltrate it via these radio signals. An attacker with appropriate hardware can receive the electromagnetic signals, demodulate and decode the data, and retrieve the exfiltrated information.*

Attacks on air-gapped networks involve multi-phase strategies to breach isolated systems by delivering specialized malware through physical media or insider agents, initiating malware execution, propagating within the network, exfiltrating data using covert channels or compromised removable media, establishing remote command and control, evading detection, and covering tracks.

In the context of the RAMBO attack, the adversary must infect the air-gap network in the initial phase. This can be done via a variety of attack vectors. An attacker could plant malware on a USB drive and physically introduce it into an air-gapped network. An unsuspecting insider or employee might connect the USB drive to a computer within the isolated network, unknowingly activating the malware and allowing it to propagate and exfiltrate data through the same USB drive or via covert channels. An insider with access to the air-gapped network might intentionally introduce malware or provide unauthorized access to external parties. This could involve transferring sensitive data to personal devices or using covert communication methods like steganography to hide data within innocent-looking files. An attacker could also compromise hardware components or software updates during the supply chain process.

I'll interrupt to note that the particular power of this attack is the degree to which its effects would be unsuspected and undetected. So an adversary might introduce their RAMBO-enabled malware into a device driver that's known to be used by and needed by the targeted system. Since no one would ever imagine that a device driver update could suddenly turn a PC into a covert short-range transmitter, the updated drivers might be delivered as part of a very careful and clean offline CD or DVD carried update. And that's all that would be required. Mordechai continues:

Once these components are installed within the air-gapped network, hidden malware might activate and communicate with external parties. Note that APTs (Advanced Persistent Threats) in the past have targeted highly secured and air-gapped networks. Recently, in August 2023, researchers at Kaspersky discovered another new malware and attributed it to the cyber-espionage group APT31, which targets air-gapped and isolated networks via infected USB drives.

In the second phase of the attack, the attacker collects information, e.g., keylogging, files, passwords, biometric data, and so on, and exfiltrates it via the air-gap covert channel.

In our case, the malware utilizes electromagnetic emissions from the RAM to modulate the information and transmit it outward. A remote attacker with a radio receiver and antenna can receive the information, demodulate it, and decode it into its original binary or textual representation.

For the actual generation of RAMBO's RF signals, he explains:

When data is transferred through a RAM bus, it involves rapid voltage and current changes, mainly in the Data bus. These voltage transitions create electromagnetic fields, which can radiate electromagnetic energy through electromagnetic interference (EMI) or radio frequency interference (RFI). The frequency range of electromagnetic emanation from the RAM bus mainly depends on its specific clock speed, measured in megahertz (MHz) or gigahertz (GHz). This clock dictates how quickly data can be transferred between the CPU and memory. The emanation levels are influenced by other bus characteristics, including its data width, clock

speed, and overall architecture. Faster RAM buses such as DDR4 and DDR5 having wider data paths can lead to quicker data transfers with increased emissions.

When data is read from or written to memory, electrical currents flow through the RAM chips and the associated traces on the printed circuit board (PCB). These electrical currents generate electromagnetic fields as a byproduct, which radiates EM energy. To create an EM covert channel, the transmitter needs to modulate memory access patterns in a way that corresponds to binary data. For instance, they could alter the timing or frequency of memory access operations to encode information. The sender and receiver must establish rules that define how memory access patterns translate to binary values. For example, reading or writing an array to the physical memory with a specific timing interval might represent a '0' while another interval represents a '1'. The receiver detects and decodes the EM emissions caused by the modulated memory activity. This could involve sensitive radio frequency (RF) receivers or electromagnetic field sensors.

One algorithm used OOK – On-Off Keying – modulation, a basic form of digital modulation used in communication systems to transmit digital data over a carrier wave. In our case, the OOK modulation involves turning the carrier wave on and off to represent binary data, where the presence of the carrier wave generated by memory activity corresponds to one binary state ("1"). The absence of the electromagnetic carrier wave (thread sleep()) corresponds to the other binary state ("0"). Note that to maintain the activity in the RAM buses, we used the MOVNTI instruction which stands for Move Non-Temporal Integer. It performs a non-temporal store of integer data from a source operand to a destination memory location. This instruction is primarily associated with optimizing memory operations for certain types of data transfers, particularly in cases where the data is not to be reused immediately. For the beginning of the transmission, we used the preamble sequence of 10101010, allowing the receiver to be synchronized with the transmitter.

For the fast transmission, we used Manchester encoding. In this encoding, each bit of the binary data is represented by a transition or change in signal level within a fixed period. Manchester encoding ensures a consistent number of signal transitions, making it useful for clock synchronization and error detection.

I smiled when I saw that Mordechai had chosen to use Manchester encoded signaling since it's extremely simple and straightforward and it's likely the best solution to his need. Manchester encoding is still in wide use today due to its simplicity and robustness, though it dates back to 1948 where it was invented and first used to store and retrieve data on the magnetic storage drum for the University of Manchester's Mark 1 digital computer.

Because Manchester encoding provides such an elegant solution to a common problem – so common that, as I noted, it's still being used today in consumer IR remote controls and RFID tags – and since it provides another interesting dip into pure communications engineering and abstract computer science, I want to take some time to examine how it works.

The problem Manchester encoding beautifully solves is known as "clocking". If you have a single bit channel, as with RAMBO where we have a radio signal that's either on or off, or a wire that's either carrying a current or not, or a remote control's infra-red LED that's either on or off, a significant problem arises when a long series of some number of '1's or '0's occurs in a sequence because the question quickly becomes, exactly how many 0's or 1's was that?

If some time passes without anything happening, for example, the radio is off or on for a while, how is the receiver to know **precisely** how many bits were just transmitted? If the sender and the receiver **both** had perfect and exactly equal knowledge of time passage, it would theoretically be possible to just count the elapsed time between a change from On to Off or Off to On, then divide that by the time per bit to determine exactly how many "bit times" had elapsed. And the guys at Manchester may have initially tried that back in 1948. But in their case, slight variations in the speed of their drum storage rotation rate would have quickly shown them that they needed something system that would be far more tolerant of slight timing variations. And even today, two clocks are never precisely synchronized nor are ever running at precisely the same rate.

Communications designers have solved this problem by creating systems known as self-clocking encoding. Self-clocking encoding systems ensure that something always happens often enough for the receiving end to stay synchronized with the sending end, even if their timing is not precise. And Manchester encoding, first used 76 years ago, does exactly that. Here's how it works:

The key to understanding any encoding is to recognize that the signal is no longer the data. Mordechai mentioned simple on-off keying which is an **unencoded** system. With simple on-off keying, the signal IS the data. But that's where we run into trouble if many '0's or '1's are sent in an uninterrupted sequence. So any encoding that we employ breaks this simple relationship between the signal and the data.

To talk about this, we'll refer to the signal as being "low" or "high" whereas the data bits are a '0' or a '1'. So "low" or "high" would mean that RAMBO's RAM transmission is either off or on.

A '1' data bit is encoded as a low followed by a high, whereas a '0' bit is encoded as a high followed by a low. In other words, RAMBO transmits a '1' bit by having its RAM transmitter first not transmitting anything, then having it transmit. And it sends a '0' by having its RAM transmitter first transmitting a signal, then switching it off and not sending any signal.

The best way to think of this is that in Manchester encoding a '1' bit is encoded as a ***transition*** from low to high whereas a '0' bit is encoded as a ***transition*** from high to low. This means that a so-called "bit cell" – which is the period of a single data bit – always contains two opposite states, both a low and a high, and the **direction** of the transition between those two states is that bit cell's data – a 0 or a 1. If the bit cell contains a transition from low to high, radio off to radio on, that's RAMBO sending a '1'. And if the bit cell contains a transition from high to low, that's RAMBO sending a '0'.

Now, if you think about this for a second you'll see a problem. In order to send a pair of '1's we need to have back-to-back low-to-high transitions – in other words, RAMBO radio off to RAMBO radio on transitions. But at the end of that first bit the radio will be on and the next '1' we're sending requires the radio to start by being off. We solve this problem by completely ignoring any inter-bitcell transitions. In other words, only the transitions occurring in the middle of bit cells carry any data. The transitions occurring in between are ignored.

So now, assuming that you've been following along carefully, you're wondering how the receiver can tell the difference between the data transitions occurring in the middle of the bit cells and the transitions we're supposed to ignore which may or may not occur in between bit cells in order to get ready for the next bit. Manchester encoding provides that answer because every bit cell **must** always contain a transition, whereas there may or may not be any transition in between two bit cells.

If you doodle with a pencil and paper for a bit you'll quickly see that any receiver can perfectly "lock onto" the location of the bit cells the very first time a transition is **missing**, since that can only be the period in between bit cells. That means that the next transition must be in the exact center of a bit cell as far as the transmitter is concerned.

So if the receiver knows only the approximate rate at which the transmitter is sending bits, that's now sufficient to allow it to judge when an inter-cell transition opportunity has passed and when the next guaranteed-to-be-present transition occurs. And when that happens, the receiver updates its self-clocking "lock" which prepares it to judge whether the next transition occurs quickly, meaning that it's an inter-cell transition, or not until it's expecting the next data bit transition.

This simple system works so well that it was used by the earliest Ethernet physical layer standards and as I mentioned earlier, it's still used today by consumer home entertainment infra-red remote controls as well as by RFID and near-field communications.

Mordechai had considered both simple On/Off keying and Manchester encoding. He wrote:

> *Our analysis shows that the Manchester encoding is more relevant for the requirements of the RAMBO covert channel due to two main reasons; (1) the encoding aids in clock synchronization between the sender and receiver, and (2) the frequent transitions make it easier to detect errors caused by signal loss, interference, or distortion. However, it's important to note that Manchester encoding doubles the required bandwidth compared to direct on/off binary encoding, as each bit requires two signal states within the bit interval.*

Okay, so how did all of this turn out? Keylogging can be exfiltrated in real-time since UNICODE is only 16 bits per keystroke. A 4096-bit RSA encryption key can be exfiltrated in 4.096 seconds and biometric information and small files such as .JPGs and small documents require a few seconds at the system's fastest speeds.

They conclude that *"This indicates that the RAMBO covert channel can be used to leak relatively brief information over a short period."*

They were also able to receive this information at a distance of up to 700 centimeters. For those of us who grew up using the Imperial system of measurement, 700 centimeters is 23 feet (!) So this is a useful and impressive feat.

I would imagine that if you got yourself a well-tuned and well-aimed Pringles can, you might be able to significantly improve on that performance distance. For at least this first round of research, they seemed less focused upon distance than feasibility. They've certainly shown that their RAMBO system is feasible.

It's been known for a long time that electronic devices generated and radiated electromagnetic interference while they were in use. The somewhat strained acronym TEMPEST stands for "Telecommunications Electronics Materials Protected from Emanating Spurious Transmissions." So, TEMPEST-hardened devices are those which incorporate specific countermeasures designed to block or mask any useful information-carrying emanations from electronic equipment.

We can hope that any air-gapped machines which have been deliberately disconnected from any traditional form of data communications will have also been shielded so that none of the noise generated by the system's motherboard is able to find its way into the surrounding environment. It would be necessary, of course, to first infect any such machine with RAMBO technology malware. But if that could be accomplished any otherwise unprotected machine could be turned into a RAMBO transmitter.