Security Now! #1015 - 03-04-25

Spatial-Domain Wireless Jamming

This week on Security Now!

Firefox amends their privacy policy — the world melts down. Signal threatens to leave Sweden. Aftermath of the massive $1.5 billion Bybit ETH heist. It turns out that it wasn't actually Bybit's fault. "The Lazarus Bounty" monitoring and management site. Mozilla's commitment to Manifest V2 (and the uBlock Origin). What does the ACM's plea for memory-safe languages mean for developers? What exactly are memory-safe languages? Australia joins the Kaspersky ban. Gmail plans to switch from SMS to QR code authentication. A SpinRite success and some fun feedback. An astonishing new technology for targeted radio jamming.

During the "phone not charging" tech support call, the customer asked "what do you mean, 'USB charger'?"

Security News

Firefox amends their privacy policy — the world melts down

By far the biggest brouhaha of the past week, at least among the circles this podcast and its faithful Firefox-using listeners move in, has been the concerns raised by Mozilla's change to Firefox's privacy policy. ArsTechnica's headline read *"Firefox deletes promise to never sell personal data, asks users not to panic"* with the follow-up: *"Mozilla says it deleted promise because "sale of data" is defined broadly."* Ars' wrote:

> *Firefox maker Mozilla deleted a promise to never sell its users' personal data and is trying to assure worried users that its approach to privacy hasn't fundamentally changed. Until recently, a Firefox FAQ promised that the browser maker never has and never will sell its users' personal data. An archived version from January 30 says:*
>
> > ***Does Firefox sell your personal data?***
> > *Nope. Never have, never will. And we protect you from many of the advertisers who do. Firefox products are designed to protect your privacy. That's a promise.*
>
> *That promise is removed from the current version. There's also a notable change in a data privacy FAQ that used to say, "Mozilla doesn't sell data about you, and we don't buy data about you." The data privacy FAQ now explains that Mozilla is no longer making blanket promises about not selling data because some legal jurisdictions define "sale" in a very broad way:*
>
> > *"Mozilla doesn't sell data about you (in the way that most people think about "selling data"), and we don't buy data about you. Since we strive for transparency, and the LEGAL definition of "sale of data" is extremely broad in some places, we've had to step back from making the definitive statements you know and love. We still put a lot of work into making sure that the data that we share with our partners (which we need to do to make Firefox commercially viable) is stripped of any identifying information, or shared only in the aggregate, or is put through our privacy preserving technologies (like OHTTP)."*
>
> *Mozilla didn't say which legal jurisdictions have these broad definitions.*
>
> *Users criticized Mozilla in discussions on GitHub and Reddit. One area of concern is over new terms of use that say, "When you upload or input information through Firefox, you hereby grant us a nonexclusive, royalty-free, worldwide license to use that information to help you navigate, experience, and interact with online content as you indicate with your use of Firefox."*

Okay. Now I'm not alarmist by nature, and I'm committed to Firefox. But Firefox is our UI portal to the Internet and the world. So, by definition, everything goes through it. Therefore, language that reads *"When you upload or input information through Firefox, you hereby grant us a nonexclusive, royalty-free, worldwide license to use that information to help you navigate, experience, and interact with online content as you indicate with your use of Firefox."* Even though I might want to, that one is a bit difficult to rationalize. I don't believe that I want any web browser, to be examining ANY of the information I input through it in ANY way for ANY purpose.

Ars' published the first edition of their report at 9:44 AM Eastern time, last Friday the 28th. They

then updated it less than an hour later, at 10:20 AM, writing:

> *Mozilla has since announced a change to the license language to address user complaints. It now says, "You give Mozilla the rights necessary to operate Firefox. This includes processing your data as we describe in the Firefox Privacy Notice. It also includes a nonexclusive, royalty-free, worldwide license for the purpose of doing as you request with the content you input in Firefox. This does not give Mozilla any ownership in that content."*

I had to reread that slowly several times. I think they're saying that in order to serve as a conduit for the information we input through Firefox they need to say something about their legal position and obligations as our information conduit. Ars continues:

> *Mozilla also took heat from users after a Mozilla employee solicited feedback in a connect.mozilla.org discussion forum. "This isn't a question of messaging or clarifying," one person wrote. "You cannot ask your users to give you these broad rights to their data. This agreement, as currently written, is not acceptable."*
>
> *Mozilla announced the new terms of use and an updated privacy policy in a blog post on Wednesday. After seeing criticism, Mozilla added a clarification that said the company needs "a license to allow us to make some of the basic functionality of Firefox possible. Without it, we couldn't use information typed into Firefox, for example. It does NOT give us ownership of your data or a right to use it for anything other than what is described in the Privacy Notice."*
>
> *One of the uses described in the privacy notice has to do with users' location data. Mozilla says it takes steps to anonymize the data and that users can turn the functionality off entirely:*
>
> > *Mozilla may also receive location-related keywords from your search (such as when you search for "Boston") and share this with our partners to provide recommended and sponsored content. Where this occurs, Mozilla cannot associate the keyword search with an individual user once the search suggestion has been served and partners are never able to associate search suggestions with an individual user. You can remove this functionality at any time by turning off Sponsored Suggestions—more information on how to do this is available in the relevant Firefox Support page.*
>
> *Some users were not convinced by Mozilla's statements about needing a license to use data to provide basic functionality. One person wrote in response to Mozilla's request for feedback: "That's a load of crap and you know it. 'Basic functionality' is to download and render web pages."*

First of all, I disagree with this disgruntled person, since "downloading and rendering web pages" is no longer all that our web browsers do for us. A perfect example of this is this sentence I'm reading right now. It's in the PDF of the show notes that was originally entered into my Firefox browser courtesy of Google Docs, an astonishing word processing system that runs in our web browsers. So it's patent nonsense to suggest that the job of today's browsers is only to download and render static web pages. Those days are long past.

I think this brings us back to the "Free Lunch Dilemma" and the reality that there's really no such thing. No one pays for or purchases the use of any web browser with their own cash. So far as I know, every web browser is "free" to use. But I have the word "free" in air-quotes, because, are our web browsers truly free?

Is it reasonable for us to expect to take and take and take from them while giving nothing in return? We want security, we want browser extension add-on stores without malware and abuse. We want absolute cross-browser compatibility and secure password storage and cross-platform operation and and and and. Who's paying for all of this? We absolutely know that maintaining a contemporary web browser is incredibly expensive. Microsoft itself was unable to do it. They gave up their independence. And the industry refuses to leave things alone. The World Wide Web Consortium, the W3C, refuses to stop moving forward with the introduction of successive advances. They want to evolve the web browser into a fully featured operating system environment. And I'm not saying that's a bad idea, because, after all, I'm editing these show notes in an astonishingly full-featured word processor which we would not have if it were not for the W3C pushing forward on features and strong standards. But this means that offering a modern state-of-the-art web browser is not only a matter of finding and fixing bugs; it also means serious never ending development to support the continually evolving standards.

The result of all this has been the creation of an incredibly capable, complex and expensive to maintain application platform that's so easy to take for granted. Mozilla's updated statement reads:

> *We still put a lot of work into making sure that the data that we share with our partners (which we need to do to make Firefox commercially viable) is stripped of any identifying information, or shared only in the aggregate, or is put through our privacy preserving technologies."*

I believe them. These are the people who said they would never sell our data. I believe that their heart is in the right place. So if, as a Firefox user, anonymity is all we can obtain from Mozilla in return for their providing us with this amazing tool for free, then I'm fine with that. That's more than we get from Google and Microsoft. What's more, I'm very appreciative and I dearly hope we never lose this alternative to being swallowed by the Chromium monster.

Signal threatens to leave Sweden

We have more bad news on the Governments -vs- Enforceable Privacy saga. Now, Sweden's government has scheduled discussions next month of legislation to require communication providers to allow police and security services access to their message content. Not surprisingly, Meredith Whittaker, Signal Foundation's president, immediately responded to this news saying that Signal will pull out of Sweden if the government there passes such a surveillance bill. In an interview on Swedish national public television, SVT, she added that such a backdoor would undermine its entire network and users across the world, not just in Sweden. And, as we know, this is the second time Meridith has indicated that Signal would leave a country over its backdoor demands. In 2023, she threatened to leave the UK if the government mandated backdoors in its Online Safety Act. And we all know that these matters are far from settled, and they need to be.

Not everyone, even in Sweden, is on the same page. It turns out that Signal is very popular and widely used within the country's armed forces, where staff were recently asked to start specifically using Signal due to its super-secure messaging capabilities. In a letter to the Swedish government, the Swedish Armed Forces wrote that the legislation could not be realized *"without introducing vulnerabilities and backdoors that may be used by third parties."* In other words, the familiar refrain that it's not possible to have it both ways. It's either secure for everyone and from everyone, or it's not truly secure for anyone.

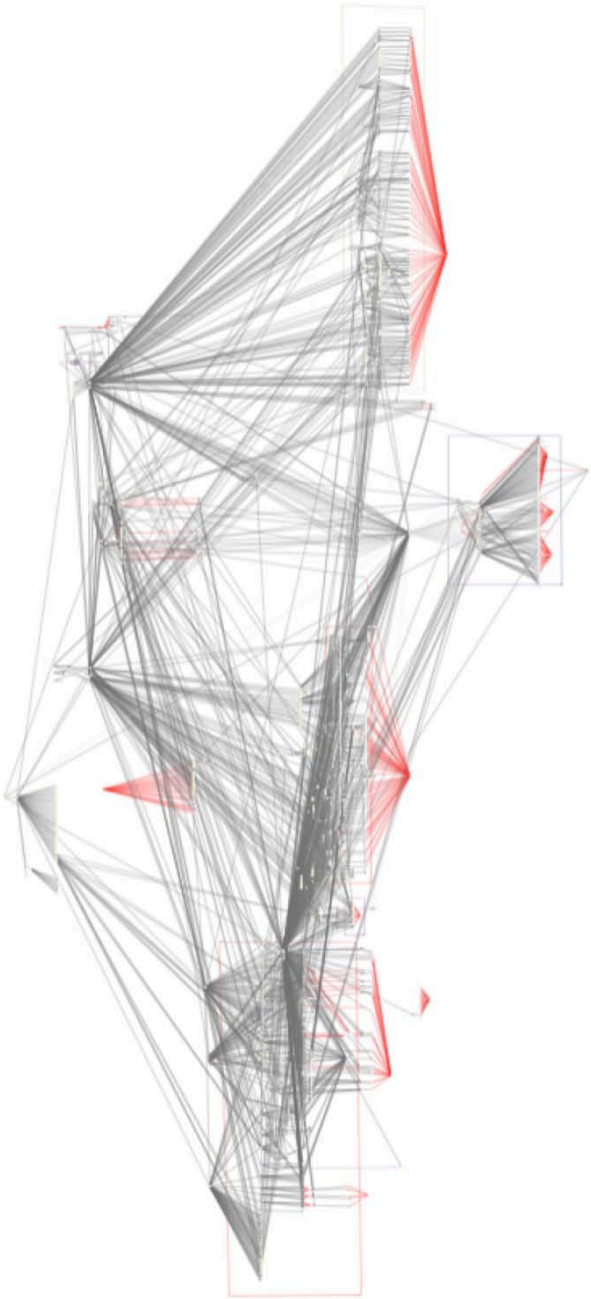
The question then, is what happens with iMessage and Google Messenger? Apple's shutting down of enabling new full end-to-end iCloud storage encryption by UK users is one thing. But what happens if Sweden mandates that **all** communications occurring within its borders be

decryptable? Just over a year ago, it was last February, when we covered Apple's announcement of their PQ3, Post-Quantum Level 3, technology for iMessage. In addition to introducing post-quantum crypto, it introduced dynamic rolling rekeying of all messages, bringing perfect forward secrecy to iMessage and bringing it fully up to the state of the art.

And now Sweden says "Sorry about that, but we've just unilaterally enacted legislation to reverse, remove and restrict the privacy rights Swedish citizens have been enjoying with their use of iMessage. We're not going to allow anyone in Sweden to enjoy the benefit of that level of security, because it makes us nervous and it might be abused."

We know what Signal will do. They've made that very clear, and they really have to follow through with their promise. But what will Apple do?

And on this topic I solicited some help from ChatGPT's o3-mini model, working with it to come up with a good acronym for this mess. So I present "NOCRYPT" — which stands for "Nationwide Outlawing of Cryptography, Restricting Your Privacy, Too."



Bybit Aftermath

Following up on last week's news of the largest ever cryptocurrency heist by North Korea, the short version is... it looks like they're going to get away with it.

I have an interconnection chart from Chainalysis which depicts the complexity of North Korea's laundering efforts so far.every endpoint node is an intermediate address with token swaps and cross-chain movements that not only attempt to obscure the stolen funds, but also serve to demonstrate the far-reaching consequences of this exploit across the broader crypto ecosystem.

So how's the recovery effort going so far? Chainalysis reports that a whopping $40 million of the $1.5 billion dollars have been recovered. So only another $1.46 billion dollars to go. Chainalysis wrote:

Despite the severity of Bybit's attack, the inherent transparency of blockchain technology presents a significant challenge for malicious actors attempting to launder stolen funds. Every transaction is recorded on a public ledger, enabling authorities and cybersecurity firms to trace and monitor illicit activities in real time.

Collaboration across the crypto ecosystem is paramount in combating these threats. The swift response from Bybit, including its assurance to cover customer losses and its engagement with blockchain forensic experts, exemplifies the industry's

commitment to mutual support and resilience. By uniting resources and intelligence, the crypto community can strengthen its defenses against such sophisticated cyber attacks and work toward a more secure digital financial environment.

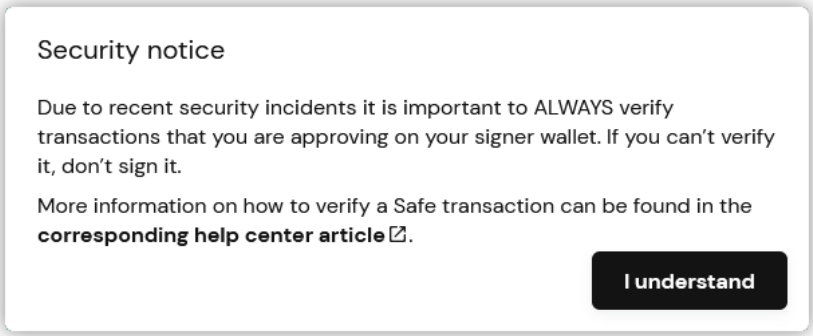
We are working with our global teams, customers, and partners across both the public and private sectors to support multiple avenues for seizure and recovery in response to this attack. Already, we've worked with contacts in the industry to help freeze more than $40 million in funds stolen from Bybit and continue to collaborate with public and private sector organizations to seize as much as possible. We will continue to provide updates on this matter.

Bybit intrusion originated at Safe{Wallet}

Meanwhile, answers to questions of how ByBit could have screwed up so much so as to lose $1.5 billion in Ethereum to North Korean hackers are beginning to trickle in. What's been learned is that the intrusion into Bybit was of their making than was originally reported. The actual intrusion originated at the supplier of one of their services, an organization named Safe{Wallet}.

Safe{Wallet} is a multisig wallet provider and the new evidence reveals that the North Korean hackers initially hacked them. The hackers injected malicious code into the Safe{Wallet} domain which selectively targeted Bybit's smart contracts and multi-sig process. Safe{Wallet} says it has now removed the code – one would hope. And also in the meantime the FBI has confirmed North Korea's involvement in the hack and linked it to a group it tracks as TraderTraitor.

Being curious about this Safe{Wallet} service, I went over to see what they were about and I was greeted by a homepage intercept which dimmed and required that I click an "I understand" button on a pop-up "Security notice" which read:



An abbreviated form of this message was repeated at the top of the page behind that front page intercept. Without digging into the weeds of all this, what we see is evidence of this newer trend of assembling a working system from many various bits and pieces of services offered by others. There's plenty of support for the "let's not reinvent the wheel" approach, and the idea of allowing specialists to focus upon their specialty where they are able to add value. This is the modern day equivalent of building apps from library components We've seen that this model can and has suffered from supply chain attacks. And now we've seen another example of a failure of the online service provider model.

Given all of the evidence we have now, I would tend to hold the Bybit guys harmless in this. This was not their fault. They trusted in the security and integrity of a service whose entire job it was to provide exactly that trusted security. The very expensive breach lies at the feet of the Safe{Wallet} service provider whose network was infiltrated and used to then perpetrate this

$1.5 billion heist upon Bybit and their depositors.

The Lazarus Bounty
Meanwhile, as we noted last week, the Bybit guys know how to motivate the Internet's bounty hunters ... to the tune of a 10% "instant payout bounty" for the recovery of any of the stolen coinage. They named this "The Lazarus Bounty" after the infamous North Korean gang who, as I just noted, the United States Federal Bureau of Investigation and other have independently confirmed was behind the theft. The Bybit guys quickly created a bounty leaderboard and payout tracking website to manage the bounty, which is this week's shortcut of the week. Check it out at https://grc.sc/1015 (https://www.lazarusbounty.com/en/).

As of Sunday evening when I'm writing this, the total available bounty is $140,000, or 10% of the estimated $1.4 billion dollars stolen. (Estimates range between $1.5 and $1.4 billion due to fluctuations in the price of Ethereum). The total aggregate awarded so far is $4,286,625, spread across 17 bounty recipients, with the largest being that $42 million mentioned earlier.

What's clear is that the $1.5 billion was almost too much value to launder. In order to keep its subsequent laundering sub-transactions from being suspicious, that $1.5 billion needed to be broken into a huge number of much smaller amounts, spread out into many wallets and then rapidly moved, broken, reassembled and further mixed. Last week I described the process as a shell game and I think that's still a good analogy.

Frankly, at this point, even though only 10 days have gone guy, as detectives apparently say, the trail is growing ever colder with each passing day. It's looking like very little of those proceeds will find their way back home. But this LazarusBounty site will allow us to keep an eye on it.

"Mozilla's approach to Manifest V3: What's different and why it matters for extension users"
Returning to the topic of Mozilla, last week, in the wake of Chrome's enforcement of their V3 browser extension manifest – and the sunsetting of V2 which forced the full strength uBlock Origin to finally and fully leave the Chrome web store – Mozilla took the opportunity to reaffirm their commitment to remaining V2-compatible. With their blog posting titled *"Mozilla's approach to Manifest V3: What's different and why it matters for extension users"*. After some prologue about the role and importance of browser extensions, they explained:

> *Right now, all major browsers — including Firefox, Chrome and Safari — are implementing the latest version of this platform, Manifest V3. But different browsers are taking different approaches, and those differences affect which extensions you can use. Principle 5 of the Mozilla Manifesto states: Individuals must have the ability to shape the internet and their own experiences on it. That philosophy drives our approach to Manifest V3.*
>
> - *More creative possibilities for developers — We've introduced a broader range of APIs, including new AI functionality that allows extensions to run offline machine learning tasks directly in the browser.*
>
> - *Support for both Manifest V2 and V3 — While some browsers are phasing out Manifest V2 entirely, Firefox is keeping it alongside Manifest V3. More tools for developers means more choice and innovation for users.*

Giving people choice and control on the internet has always been core to Mozilla. It's all about making sure users have the freedom to shape their own experiences online.

Google began phasing out Manifest V2 last year and plans to end support for extensions built on it by mid-2025. That change has real consequences: Chrome users are already losing access to uBlock Origin, one of the most popular ad blockers, because it relies on a Manifest V2 feature called ***blockingWebRequest****.*

Google's approach replaces ***blockingWebRequest*** *with* ***declarativeNetRequest****, which limits how extensions can filter content. Since APIs define what extensions can and can't do inside a browser, restricting certain APIs can limit what types of extensions are possible.*

Firefox, will continue supporting both blockingWebRequest and declarativeNetRequest — giving developers more flexibility and keeping powerful privacy tools available to users.

We pretty much knew this was what Mozilla planned. But it's nice to have Mozilla's intent made clear. And with the Internet becoming ever more important and websites unfortunately ever more insistent upon monetizing our presence, it's increasingly important to have a tool like uBlock Origin that's able to return to us a modicum of control.

The ACM makes a plea for memory-safe languages

The ACM is the Association for Computing Machinery. Its founding in 1947 makes it not only the world's largest scientific and educational computing society, but also the oldest. It's a non-profit professional membership group with nearly 110,000 student and professional members, based in New York City. The ACM publishes over 50 journals including the prestigious Journal of the ACM, and two general magazines for computer professionals, the Communications of the ACM, also known as just "Communications" or CACM. The ACM's motto is "Advancing Computing as a Science & Profession."

The February issue of the Communications of the ACM, in its "Security & Privacy" section, contained an article titled: *"It Is Time to Standardize Principles and Practices for Software Memory Safety"*. The article was co-authored by twenty-one professionals spanning academia and industry, with expertise in memory-safety research, deployment, and policy. In it, they argue that standardization is an essential next step to achieving universal strong memory safety.

I'm just going to share the introduction of this very long, detailed and well thought out editorial:

For many decades, endemic memory-safety vulnerabilities in software trusted computing bases (TCBs) have enabled the spread of malware and devastating targeted attacks on critical infrastructure, national-security targets, companies, and individuals around the world. During the last two years, the information-technology industry has seen increasing calls for the adoption of memory-safety technologies, These have been framed as part of a broader initiative for Secure by Design, from government, academia, and within the industry itself These calls are grounded in extensive evidence that memory-safety vulnerabilities have persistently made up the majority of critical security vulnerabilities over multiple decades, and have affected all mainstream software ecosystems and products — and also the growing awareness that these problems are almost entirely avoidable by using recent advances in strong and scalable memory-safety technology.

In this Inside Risks column, we explore memory-safety standardization, which we argue is an essential step to promoting universal strong memory safety in government and industry, and, in turn, to ensure access to more secure software for all.

During the last two decades, a set of research technologies for strong memory safety — memory-safe languages, hardware and software protection, formal approaches, and software compartmentalization — have reached sufficient maturity to see early deployment in security-critical use cases. However, there remains no shared, technology-neutral terminology or framework with which to specify memory-safety requirements. This is needed to enable reliable specification, design, implementation, auditing, and procurement of strongly memory-safe systems. Failure to speak in a common language makes it difficult to understand the possibilities or communicate accurately with each other, limiting perceived benefits and hence actual demand. The lack of such a framework also acts as an impediment to potential future policy interventions, and as an impediment to stating requirements to address observed market failures preventing adoption of these technologies. Standardization would also play a critical role in improving industrial best practice, another key aspect of adoption.

This Inside Risks column is derived from a longer technical report published by the same authors, which includes further case studies and applications, as well as considering the potential implications of various events and interventions on potential candidate adoption timelines.

It's easy to read between the lines here. What's being said is that we need to establish a common and universally agreed upon framework and terminology so that both public government and private commercial sector purchasers of next-generation network and security technology will have some actionable means for specifying in their requests for quotes, bids and purchasing contracts that every component of the system has been developed in, and is using, **only** memory-safe language technologies.

In other words, the writing is on the wall ... and that writing says that the time is now for anyone who may have ambitions to sell their future products to government or large enterprises to begin the process of rewriting them from scratch in approved memory safe languages. I can 100% guarantee that future purchasing requirements documents will be specifying that only appliances that have been written in pure memory safe languages will be considered for purchase, and that if any problem should later occur and it turn out that the proximate cause of the trouble was the use of non-memory safe languages, the supplier will be held responsible for the damages due to their having made substantial fraudulent misrepresentations.

There's a great web site "https://www.memorysafety.org/docs/memory-safety/", created by the ISRG, the Internet Security Research Group (ISRG). They explain:

Our first goal is to move the Internet's security-sensitive software infrastructure to memory safe code. Many of the most critical software vulnerabilities are memory safety issues in C and C++ code. While there are ways to reduce the risk, including fuzzing and static analysis, such mitigations do not eliminate the risk and they consume a lot of resources on an ongoing basis. Using memory safe languages eliminates the entire class of issues. We recognize the amount of work it will take to move significant portions of the Internet's C and C++ software infrastructure to memory safe code, but the Internet will be around for a long time. There is time for ambitious efforts to pay off. By being smart about our initial investments, focusing on the most critical components, we can start seeing significant returns within 1-2 years.

Our second goal is to change the way people think about memory safety. Today it's considered perfectly normal and acceptable to deploy software written in languages that are not memory safe, like C and C++, on a network edge, despite the overwhelming evidence for how dangerous this is. Our hope is that we can get people to fully recognize the risk and view memory safety as a requirement for software in security-sensitive roles.

This effort is called "Prossimo" and is being funded by contributions from Google, AWS, CISCO, the Sovereign Tech Fund, Craig Newmark Philanthropies, Chainguard, Cloudflare, Shopify and others. Their current initiatives include: An implementation of TLS in the Rust language where they say: "Let's get the Rustls TLS library ready to replace OpenSSL in as many projects as possible." Of the Linux project they write: "Let's make it possible to write memory safe drivers for the Linux kernel." There's a project called Hickory which will be a memory safe, high performance, fully recursive DNS resolver (and that one is nearly ready). The AV1 project is creating a fully memory-safe AV1 decoder to deliver great performance. There's a project to develop a high-performance memory-safe zlib compression library. Of their SUDO project they say: "Let's make the utilities that mediate privileges safer." And they have similar initiatives for NTP, Apache, curl and various other tools.

So if the future is memory-safe languages, which ones are those? The MemorySafety.ORG site has a page asking and answering "What is Memory Safety?" What I appreciated was that they perfectly summarized this in just two sentences, writing:

Memory safety is a property of some programming languages that prevents programmers from introducing certain types of bugs related to how memory is used. Since memory safety bugs are often security issues, memory safe languages are more secure than languages that are not memory safe.

It could not be said more clearly and succinctly than that. *"Memory safe languages are more secure than languages that are not memory safe"* Okay. So what languages? They continue with their answer and explanation, writing:

Memory safe languages include Rust, Go, C#, Java, Swift, Python, and JavaScript. Languages that are not memory safe include C, C++, and assembly.

To begin understanding memory safety bugs, we'll consider the example of an application that maintains to do lists for many users. We'll look at a couple of the most common types of memory safety errors that can occur in programs that are not memory safe.

Out of Bounds Reads and Writes *(also known as a buffer overflow)*

If we have a to do list with ten items, and we ask for the eleventh item, what should happen? Clearly we should receive an error of some sort. We should also get an error if we ask for the negative first item.

Under these circumstances, a language that is not memory safe may allow a programmer to read whatever memory contents happen to exist before or after the valid contents of the list. This is called an out of bounds read. The memory before the first item of a list might be the last item of someone else's list. The memory after the last item of a list might be the first item of someone else's list. Accessing this memory would be a severe security vulnerability! Programmers can prevent out of bounds reads by diligently checking the index of the item

they're asking for against the length of the list, but programmers make mistakes. It's better to use a memory safe language that protects you and your users from the class of bugs by default.

In a memory safe language we will get an error at compile time or a crash at run time. Crashing the program may seem severe, but it's better than letting users steal each others' data!

A closely related vulnerability is an out-of-bounds write. In this case imagine we tried to change the eleventh or negative first item in our to do list. Now we are changing someone else's to do list!

Use After Free

Imagine we delete a to do list and then later request the first item of that list. Clearly we should receive an error, as we shouldn't be able to get items from a deleted list. Languages that are not memory safe allow programs to fetch memory that they've said they are done with, and that may now be used for something else. The location in memory may now contain someone else's to do list! This is called a use-after-free vulnerability.

So, how common are memory safety vulnerabilities?

*Extremely. A recent study found that **60 to 70% of vulnerabilities in iOS and macOS** are memory safety vulnerabilities. Microsoft estimates that 70% of all vulnerabilities in their products over the last decade have been memory safety issues. Google estimated that 90% of Android vulnerabilities are memory safety issues. An analysis of 0-days that were discovered being exploited in the wild found that more than 80% of the exploited vulnerabilities were memory safety issues.*

The Slammer worm from 2003 was a buffer overflow – an out-of-bounds write. So was WannaCry (out-of-bounds write). The Trident exploit against iPhones used three different memory safety vulnerabilities (two use-after-frees and an out-of-bounds read). HeartBleed was a memory safety problem (out-of-bounds read). Stagefright on Android (two out-of-bounds writes). The Ghost vulnerability in glibc? You betcha (out-of-bounds write).

These vulnerabilities and exploits, and many others, are made possible because C and C++ are not memory safe. Organizations which write large amounts of C and C++ inevitably produce large numbers of vulnerabilities that can be directly attributed to a lack of memory safety. These vulnerabilities are exploited, to the peril of hospitals, human rights dissidents, and health policy experts.

Using C and C++ is bad for society, bad for your reputation, and it's bad for your customers.

Or in other words... "Bad!" There's just a little more that I think is worth sharing:

What other problems are associated with languages that are not memory safe?

Languages that aren't memory safe also negatively impact stability, developer productivity, and application performance.

Because languages that are not memory safe tend to allow for more bugs and crashes, application stability can be greatly impacted. Even when crashes are not security sensitive they are still a very poor experience for users.

Worse, these bugs can be incredibly difficult for developers to track down. Memory corruption can often cause crashes to occur very far from where the bug actually is. When multi-threading is involved, additional bugs can be triggered by slight differences in which thread runs when, leading to even more difficult to reproduce bugs. The result is that developers often need to stare at crash reports for hours in order to ascertain the cause of a memory corruption bug. These bugs can remain unfixed for months, with developers absolutely convinced a bug exists, but having no idea of how to make progress on uncovering its cause and fixing it.

Finally, there is performance. In decades past, one could rely on CPUs getting significantly faster every year or two. This is no longer the case. Instead, CPUs now come with more cores. To take advantage of additional cores, developers are tasked with writing multi- threaded code.

Unfortunately, multi-threading exacerbates the problems associated with a lack of memory safety. As a result, efforts to take advantage of multi-core CPUs are often intractable in C and C++. For example - Mozilla had multiple failed attempts to introduce multi-threading into Firefox's C++ CSS subsystem before finally (successfully) rewriting the system in multi-threaded Rust.

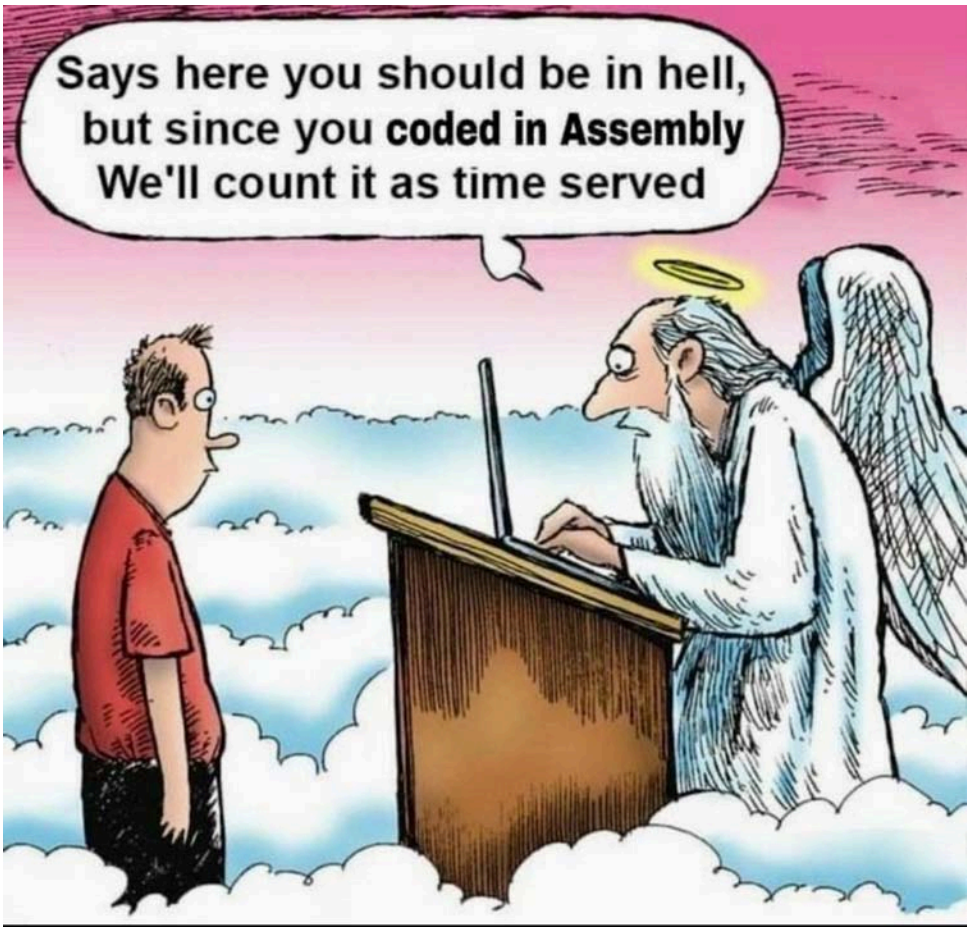
So, what's the right path forward?

Use memory safe languages! There are lots of great ones to choose from. Writing an operating system kernel or web browser? Consider Rust! Building for iOS and macOS? Swift's got you covered. Network server? Go's a fine choice. These are just a few examples, there are many other excellent memory safe languages to choose from (and many other wonderful use case pairings!).

Changing the programming language your organization uses is not something to be undertaken lightly. It means changing the skills you're looking for when you hire, it means retraining your workforce, it means making the time to rewrite large amounts of already apparently working code. Nevertheless, we believe in the long term this is required.

I took the time here to share this because I know from the feedback I receive that we have listeners who are wondering about their own paths forward. The points about application stability mean that memory-safe languages are not only more secure, but they also inherently create more stable, easier to debug and easier to maintain solutions and products.

We all know that my own native programming language is assembler, which is essentially the machine's native language with absolutely no guardrails. It would be really interesting to talk to some other truly hardcore coders, who are as fluent with assembler as I am, because my feeling is that C and C++ are dramatically MORE DANGEROUS than raw assembler itself. This is because C's entire design goal was to be as absolutely low level as possible and just barely enough above the actual machine so as to obtain machine independence. The result is that the C compiler may not do what its programmer expects. In a way, I think this makes C far more dangerous than assembler, where there's no middle man to mess things up.



So it would be interesting to see whether other assembly language coders feel the same way. One thing we know is that what I produce in assembly language tends to be far more bug free than other code we commonly encounter.

So the significant takeaway here should be a recognition that the only thing that's keeping unsafe and net productivity-inefficient languages like C and C++ going today, is inertia. Every listener of this podcast is well aware of what a powerful force inertia can be. We might even label it the main governing force. And I'm in its grip myself. I am never dropping my use of assembly language. But I'll be 70 years old in about three weeks, so I'm far closer to being done than I am to starting out. My serious advice to anyone who is closer to starting out would be to seriously consider grabbing a development environment for Rust or Go or Swift or Python and spend some time becoming very comfortable with one or more of those next-generation memory safe languages. Java is also a very strong language for internal enterprise development and a huge amount of code that's written is not aimed out to the world but is used inside the enterprise. Those are nice safe jobs if you can land one.

There really has been a change here, so you'll want to add comfort with a few of those next-generation languages to your resume.

Australia joins the Kaspersky ban

Just a quick note that the Australian government has now banned the use of Kaspersky products on government systems. All Australian government agencies must uninstall any existing Kaspersky software by April 1st. Government officials said that the software poses an unacceptable security risk to Australian government networks, opening it to foreign interference, espionage, and sabotage. It's not fair. As we know, there's been no credible evidence shown of

any wrongdoing on Kaspersky's part. And they remain valuable contributors to global security. But, they're Russian, so they're being painted with the same brush. And while it might not be fair, it's understandable.

Gmail to drop SMS in favor of QR codes

From reporting by Forbes that was picked up by ZDNet and pretty much everyone else, we learn that Google's Gmail will be dropping their historical use of less-than-super-secure 6-digit SMS codes for multi-factor authentication, replacing them with QR codes. So, rather than asking a user to enter a code received via text message, users wishing to login will be presented with a QR code to scan with their phone.

But it's unclear to me how this would work, exactly. The original text messaging solution relied upon users having their phone number pre-registered with their account. So their ability to receive a random code at that number was meant to serve as proof of their control over that pre-registered phone number and, by extension, the handset that number is currently associated with. In the parlance of multi-factor authentication, this would add an additional factor – the "something you have" factor – to the username and password which provide "something you know". The problem with a strong reliance upon text messaging is that, as we all know, our telecommunication systems are not secure in the face of outright hacking or various SIM swapping schemes that can and have been used to intercept text messages. But as I said, what's unclear to me is how presenting the user with a QR code solves this problem.

The reporting on this says that using a QR code prevents someone from being tricked into revealing the 6-digit code they've just received. Okay. So that on-screen QR code presumably contains a webpage link with a bunch of crypto-crap in its URL? That's very fancy, but it's unclear what prevents a bad guy who's trying to login, from receiving that QR code themselves and then scanning it with their phone? What makes that any more secure? Thinking that I must be missing something, I checked around and found that (a) this is such big news that everyone else is reporting it too, that (b) everyone is just repeating the same information from the one Forbes guy, and that (c) the very few people who have stopped to ask exactly how this would work have the same questions I have. Just waving our arms around and saying *"QR codes instead of SMS codes"* does not a secure login protocol make. Many sites are screaming that having Gmail using QR codes makes the situation worse since users cannot natively read QR codes, so they could be used to get up to all manner of mischief.

But stepping back from the hysteria over this, for a user to authenticate securely with an additional physical factor, that physical factor must be something that an attacker cannot also have. This is what has made secure physical tokens the go-to solution when maximum security is required. But a generic smartphone does not fill that bill. The only way I can see this working, would be for future Gmail users to also have some sort of synchronized Gmail authentication application running in their smartphones. That application would receive the QR code to close the authentication loop. And, yes, I know, that does sound suspiciously like the technology I originally developed, documented and demonstrated in Sweden, and in Ireland, and here on TWiT many years ago. The SQRL technology essentially created a physical software token in its user's phone, using a QR code to close the loop. So it'll be interesting to see if Google follows in SQRL's footsteps in that regard, too. And you know what they say about imitation and flattery. There may be some flattery coming my way before long.

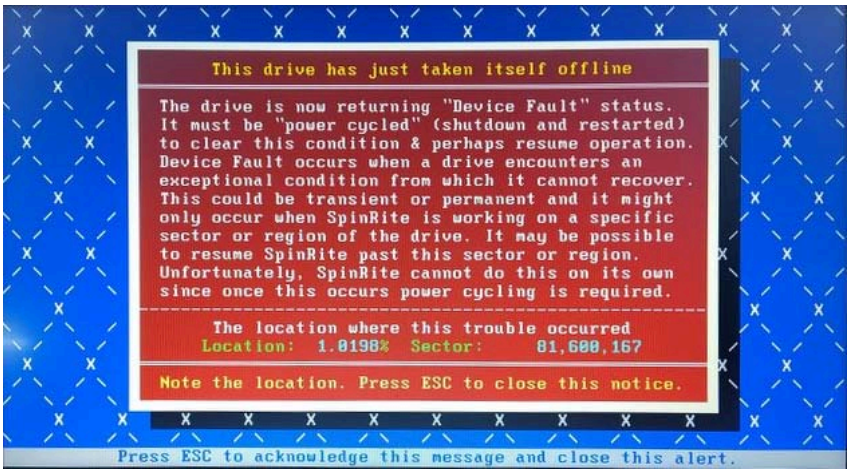
SpinRite

And speaking of flattery, a recent podcast listener of ours, Mattias Dewulf, is about to hear me share the experience he wrote to us about, after purchasing his first copy of SpinRite in desperation. He gave his email the subject line "Success Story (Level3) Dead Kingston SSD", and started off writing:

> *I own a portable Kingston XS2000 USB-C 4TB drive to store my backups.*

He included a link (which I have in the show notes for anyone who may be interested), and I was surprised by the small size of the drive's package. It's a lovely little drive that's available in 500GB, 1, 2 and 4 terabyte capacities. And as might be expected, the 4TB version is a bit pricey. It can probably be purchased online for less than the suggested retail, but Kingston's site lists the 4TB drive at £272.88 which is about $350 US dollars. So when a little drive like this dies, it's not something you want to give up on. And die, it had. He explained:

> *I configured the drive with 2 partitions: 2TB for Linux (luks encrypted ext4), 2TB for Windows (nfts, bitlocker to go). The drive recently started throwing nasty errors when trying to read files from it.*
>
> *I first noticed issues when I was working on the Linux partition. While copying a file, the copy operation stalled and the drive completely disappeared from the operating system (HP Omen laptop running Ubuntu 24.04 Cinnamon). (See messages output at the bottom). At first I thought it was perhaps a usb bus error or a bad cable. But the issue persisted and I started seeing file copy errors with Explorer hangs and USB disconnects on my Windows 11 OS while working on the Windows partition, I got really worried and started investigating.*
>
> *I could reproduce the errors on several laptops and different USB cables and ports. Some files simply were unreadable and caused the drive to disappear from the OS. All evidence pointed to an issue with the drive itself. It had become completely unusable and recovering files was a nightmare (1 by 1, keeping track of the ones that killed the drive).*
>
> *I knew of the existence of GRC.COM (ShieldsUP!) and Spinrite since somewhere in the nineties. And I started listening to the Security Now podcasts about a year ago, because I started running (and got really bored while running for hours). And during the runs I also heard your stories about the positive effect of Spinrite level3 runs on the consequences of the 'read disturb' problem that affects SSDs. I put 1 and 1 together and suspected this drive might contain a controller that handles the "tough" slow reads badly and dies...*
>
> *Sidenote: I never had the need for Spinrite. I was always able to recover my data using Open Source Linux tools. And believe me, I have done a lot of recovery.... (don't tell anyone that you know a thing or two about computers, they will find you, with their unreadable disks or NAS appliances).*
>
> *But as I lost access to the disk with other tools, I bought a copy of Spinrite (I figured it was also a way to support your work). So I went ahead and ran Spinrite against the Kingston drive. Level 2 reads also killed the disk and made it go offline. Repeated runs killed it every time at the same percentage and more or less the same sector:*



So I moved on and tested a partial level 3. I interrupted it at 1% to see if the level2 read would make it further on the disk afterwards. And behold, it did. This time the level2 read died right after the 1% of data I rewrote using the level3 scan. So I let it run for 3 days across the full 4TB disk.

The drive was rewritten completely and no errors were found during the level3 scan. I was able to read all my files afterwards, both on Linux and Windows. I am amazed and still trying to understand what your tool is doing differently. I suspect it might be something in the 'read a sector/write the same sector' logic and the lower speed it does it at?

I am starting to hate SSD technology more and more. Its only advantage is speed. But the industry has done so many bad things and compromised to try and reduce the cost. I had my fair share troubleshooting SSD issues. The most memorable one is probably my bug report to Kingston about their SV100S2 drives. It took me 6 months to convince them their SSD died after 126 days of uptime (after cold boot). It took them a long time to believe me and then discover a 32-bit overflow in the SSD controller firmware.

His email provided a link to a Kingston Release Notes PDF where he quotes is saying: "Resolves an issue where the drive becomes unresponsive after continuous usage for 2,982 hours and 37 minutes without power cycle. Issue does not occur if drive is power cycled prior to the 2,982 hour limit". And his note concludes:

In any case, I owe you a beer or two... Kind regards, Mattias

So, I have a couple of thoughts.

First of all, Mattias, I consider our books completely balanced here. You owe me nothing, though I'd be glad to share some beers. You purchased a copy of SpinRite, which does indeed allow me to afford to keep GRC on the air and to keep various GRC products alive and moving forward. The revenue from the sales of my software also serves to remind my wonderful wife that I'm not completely insane to be spending the majority of my time working on software. That was the deal we made when we met – she knew what she was in for – but a bit of positive reinforcement goes a long way. So thank you for that.

You've also offered a textbook perfect use case for SpinRite. The only part of your story that made me grimace was that 3 days were required for a full drive rewrite of a 4TB USB-connected drive. I'm sure this was largely due to SpinRite still being hosted by DOS. The performance

improvement for USB-connected drives will be one of the biggest benefits offered by SpinRite 7 because it will run natively under Windows or WINE. Occasionally rewriting entire SSD drives is so beneficial for their health that I'm eagerly looking forward to the day when doing so will be more practical. Even better, will be SpinRite's ability to surgically locate and rewrite only the "slow spots" of SSDs that have become troubled. But one step at a time.

The other comment I had was that I have come to feel exactly as Mattias has about SSDs. They are screamingly fast, but they cannot be relied upon. I've switched every one of GRC's servers, which were initially all SSD, back to using spinning drives exclusively. Every one of the SSDs I was using eventually died. And I had purchased the highest quality, modest size, most reliable single level cell SSDs available. No data has ever been lost since even the SSDs were running in a RAID 6 configuration with a full two drives of redundancy. I would never run any mission-critical drive solo. The non-RAIDed SSDs I use are automatically backed up to Synology NAS boxes which all have spinning disks with maximum two-drive redundancy, and the working directories where I spend my days are being continuously backed up with SyncThing.

These days, mass storage is just too inexpensive to not plan for its failure. But when something does eventually die, as happened with Mattias' cute little 4TB backup drive, as long as I'm alive I expect that SpinRite will be there to save the day and will only be getting better at doing so.

Listener Feedback

Josh Fenton

Hi Steve: In the last episode you mentioned that Apple will, after some date, disable GDP for Apple Users in the UK. How would this actually be possible? If the only thing that Apple's servers possess is an encrypted blob of data without the key to decrypt it, then wouldn't it be impossible for Apple to unilaterally revert users' encrypted data back into plaintext? I can see how they could simply delete the blob, but with syncing across devices enabled this would result in massive data loss for its users. Thanks, Josh Fenton

I was sure of the answer, but I went over to Apple Support to check. Under the topic *"How to turn off Advanced Data Protection for iCloud"* Apple writes: *"You can turn off Advanced Data Protection at any time. Your device will securely upload the required encryption keys to Apple servers, and your account will once again use standard data protection."*

So this is something that can be done by the user. This also suggests that a future update to iOS and macOS will enable the OS to inform its user that Apple's Advanced Data Protection feature is being withdrawn from the UK, and that after acknowledging this notice, ADP will be disabled globally for their account. At that point, every one of the user's logged-in devices will disable its local ADP setting and revert to traditional non end-to-end iCloud storage.

A listener requesting anonymity:

Viscount Systems' Freedom access control now secures the U.S. Department of Homeland Security, which uses the physical security system in dozens of field offices of Citizenship and Immigration Services (USCIS), the department's largest agency.

Oh, that's just great. As we'll recall from last week, this is the ridiculously insecure access control system that publishes its default username and password with easily ignored instructions that they should be changed. Let's hope that whoever deployed these systems for the U.S. DHS

set them up properly. And let's hope that they do not have their web portal exposed, even if it appears to be protected by an unknown username and password.

Billy Suratt / @surattb

> *What was that company you talked about on SN within the last couple years with a subscription service offering really slick Windows patching in memory?*

Ah. That would be 0patch.com and I'm glad that Billy brought them up again. I went over there to take a look around and visited their "media" page: http://0patch.com/media.html where their truly great press coverage demonstrates that these guys likely have a bright future. Among the large number of articles listed as you scroll that page is an example from PC Magazine with the headline: *"End of Support for Windows 10? Not So Fast"*. It's worth reminding everyone of this, so I'll share what PC Magazine wrote:

> *For $27 per year, Slovenia-based 0patch will keep your Windows 10 machine up to date with critical security patches for up to five years once Microsoft ends support.*
>
> *As Microsoft prepares to end support for Windows 10, a third-party service is ready to step into the void by offering five years of extra updates for the popular OS. The offer comes from Slovenia-based 0patch, which has made a business out of patching out-of-date Windows operating systems, including Windows 7. It now plans on supplying critical security patches to Windows 10 once Microsoft officially stops supporting the OS in 2025.*
>
> *The service wrote in a blog post: "With 0patch, you will be receiving security 'micropatches' for critical, likely-to-be-exploited vulnerabilities that get discovered after October 14, 2025."*
>
> *The catch is that you'll have to trust 0patch, an unofficial Microsoft service, to safely maintain your Windows 10 installation. And extended support will cost €24.95 ($27) per year. Still, the price might be a bargain. Microsoft will also offer extended security updates, but the cost to business customers starts at $61 per device per year and doubles every consecutive year for up to three years. It has not announced pricing for consumers yet, but it'll likely be more than $27 per year, as Microsoft would prefer you upgrade to Windows 11. The 0patch service may appeal to those with one of the estimated 240 million PCs that are not compatible with Windows 11.*

Leo, it turns out that hardware compatibility is not my problem. After extensive testing, I've conclusively determined that, as it turns out, ***I*** am not compatible with Windows 11. And it turns out I'm not alone. PC Mag finished their report by further quoting from a 0patch blog posting:

> *"Many of us don't want to, or simply can't, upgrade to Windows 11. We don't want to because of increasing enshittification including bloatware, Start Menu ads, and serious privacy issues. We don't want to have an automated integrated screenshot- and key-logging feature constantly recording our activity on the computer."*

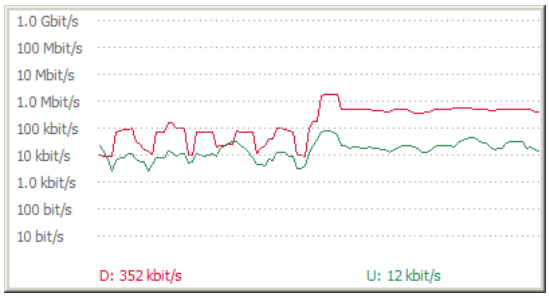
In any event, 0patch are certainly not new kids on the block. They've been offering this service, and we've been covering them on the podcast, since they first appeared many years ago. It's going to be interesting to see how Microsoft deals with the willful termination of free security fixes for problems of their own making in their own software and what they plan to charge end users who wish to continue receiving them. Thanks for the question, Billy.

David Thompson

Had a question, what network monitoring software are you using? I have seen in the past, during the DoS of GRC, you look up at the top left corner to see status'. Just curious if you had any to share.

David's quite observant. Because my primary servers are running Windows, I've taken to just using the built-in PerfMon app which monitors the servers' performance counters. And Windows allows this to be done remotely. But doing that is definitely not safe. This would normally mean exposing Windows port 445 to the public Internet – which would be begging for a visit from a hostile foreign power. This might be abbreviated OMDB, which, in this case, would stand for Over My Dead Body. So I've arranged to have secure access to the Windows performance counters of my remote servers without exposing any ports to the Internet.

But, this is a good opportunity for me to mention my very favorite LAN monitoring tool. I use and depend upon it at both of my locations. It is SO HANDY for keeping track of the WAN side of my Internet connections. The tool is called "NetWorx" from a company called SoftPerfect.com, and I've talked about it before: https://www.softperfect.com/products/networx/



I just took a snapshot of its little perfect network monitoring window which I always have up. The RED trace is incoming traffic. So if I or anyone in the house is downloading something it will jump up. And GREEN is the outgoing traffic. If I do something like save a large file that's being mirrored to my local NAS, that will happen. Then a few seconds later SyncThing will detect the local change and reach out to the other NAS to clone this changed file there. So I'll notice a jump in the green outgoing bandwidth line while the file is sent. The author of this tool also know that a logarithmic scale is what's needed to make this sort of chart useful, so that's an option.

But the coolest thing about this tool, which has a bazillion features galore, most of which I don't care about, is that it's monitoring my router rather than this PC. That's why I'm able to see the NAS using my network's outgoing bandwidth. So this is a chart of my aggregate LAN traffic, which is the same as the WAN traffic. Anyway, I really love it. It's comforting to be able to keep an eye on what's going on. You can grab it and try it for 30 days before deciding whether it's worth $15 to own forever. And while you're over there at softperfect.com look around. The company was founded in the year 2000 in Brisbane, Australia and from what I've seen they're doing great work. Something else that might be of interest is a free browser cache relocator. This can be done manually, but this little freebie makes it easy. In their description of the app they write:

Internet browsers intensively use a folder on your hard disk for temporary data — browser cache. There are various reasons why some users want to relocate this folder. For example, moving the cache to a RAM disk can speed up browsing, offload the HDD, or reduce the

wear-and-tear of the SSD.
SoftPerfect Cache Relocator is a quick and easy way to move your browser cache. This utility is intended to be used in conjunction with SoftPerfect RAM Disk, which offers all the benefits of creating disks in RAM: increase in computer performance, mitigation of the physical disk's wear-and-tear, and reduction of file system fragmentation.

You could use their RAM disk, but there are free Windows RAM disk utilities. I looked into this extensively when I was tackling the challenge of on-the-fly signing of SpinRite customer downloads since I thought I might need to be using a RAM disk as an intermediate store. The best I've found after a lot of looking and experimentation is https://imdisktoolkit.com/ the IM Disk Toolkit. It's also hands down the best tool for mounting disk images under Windows. I have links to all this in the show notes.

Alfred Deisinger

Hi Steve, I just received this notice. After 31 years, Entrust is out of the CA business.
Best regards,

And Alfred thoughtfully attached the announcement email he had received. The announcement was brief and to the point:

Dear Public Certificate Customer, We are writing to provide you with formal notice of the sale of the Entrust public certificate services business to Sectigo Limited. We provided you with initial notification by email on January 29, 2025.

Recall that we spent some time last summer looking closely at the egregious behavior of these clowns. They're the ones who blew off the warnings of the CA/Browser forum after several certificate mis-issuance events, even continuing to mis-issue new certificates after the problem had been pointed out to them and flagrant violation of the CA/Browser forum rules. This caused those who monitor such things to take a much closer look into Entrust's past conduct, and that, in turn, caused Google's Chrome browser team to decide that any of their certificates after last October 31st – Halloween – would no longer be trusted by Chrome.

Despite their promises to fix this, we predicted that this was not a survivable event and that appears to be the case. Selling their business to Sectigo effectively means that the Entrust customer base will be acquired by Sectigo. And remember that Sectigo themselves have a somewhat checkered past, since they renamed themselves after ruining the reputation of "Comodo" which was their previous name.

In any event, not being an Entrust customer, I missed this bit of under-the-radar news, so thanks again, Alfred.

"harry"

"FREEDOM Administration Login"? shouldn't that be "FREEDOM Administration INSECURE Login"? — or "FAIL" for short. It sure is a MAJOR FAIL.

I TNO, therefore I am.

Spatial-Domain Wireless Jamming

Page: https://arxiv.org/abs/2402.13773 PDF: https://arxiv.org/pdf/2402.13773

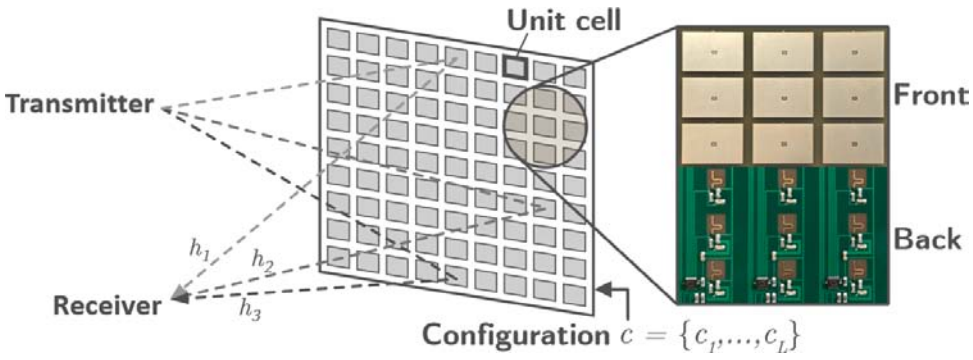
Everyone who's been following this podcast for a while knows that the way to my heart is through technical research papers. Nothing beats going to the source and hearing from the researchers who actually did the work. So when I saw this work from a team of German academics which was presented during last week's Network and Distributed System Security (NDSS) Symposium 2025, was held in San Diego, California, I knew that I needed to at least put it on everyone's radar. I think it was probably the paper's catchy title "Spatial-Domain Wireless Jamming with Reconfigurable Intelligent Surfaces." That's a crowd stopper.

Listen to what these guys explain in their paper's Abstract:

Wireless communication infrastructure is a cornerstone of modern digital society, yet it remains vulnerable to the persistent threat of wireless jamming. Attackers can easily create radio interference to overshadow legitimate signals, leading to denial of service. The broadcast nature of radio signal propagation makes such attacks possible in the first place, but at the same time poses a challenge for the attacker: The jamming signal does not only reach the victim device but also other neighboring devices, preventing precise attack targeting.

***In this work,** we solve this challenge by leveraging the emerging reconfigurable intelligent surface (RIS) technology, for the first time, for precisely targeted delivery of jamming signals.*

In particular, we propose a novel approach that allows for environment-adaptive spatial control of wireless jamming signals, granting a new degree of freedom to perform jamming attacks. We explore this novel method with extensive experimentation and demonstrate that our approach can disable the wireless communication of one or multiple victim devices while leaving neighboring devices unaffected. Notably, our method extends to challenging scenarios where wireless devices are very close to each other: We demonstrate complete denial-of-service of a Wi-Fi device while a second device located at a distance as close as 5 mm – that's 1/5th of an inch – remains unaffected, sustaining wireless communication at a data rate of 25 Mbit/s. Lastly, we conclude by proposing potential countermeasures to thwart RIS-based spatial domain wireless jamming attacks.



The illustrative photo that accompanies their paper, which I've copied into the show notes, shows a grid of antennas. This immediately suggests that they've created a 2D steerable beam, jamming transmitter, using a phased grid array.

That would be an entirely reasonable conclusion and it would be wrong. If that's what these guys had done it would be a nice piece of work, but by now it could hardly be novel.

What is novel here, is that this panel does not, itself, transmit anything. It is entirely passive and reflective; it is selectively reflective. And what's somewhat astonishing is that something that's essentially a passive reflector of WiFi or other radio signals can arrange to selectively target and deny an active WiFi device located some significant distance away from functioning. This is the sort of cool next-generation cyber spy-tech that the NSA and CIA will want to immediately set up in a lab to fully explore. It is just too cool. Here's what the inventors of this explain:

Wireless communication systems are ubiquitous and seamlessly provide connectivity to the smart and interconnected devices that permanently surround us. In our modern daily lives, we frequently use instant messaging, media streaming, health monitoring, and home automation – all of which rely on wireless systems and their constant availability. However, wireless systems utilize a broadcast medium that is open to everyone, inherently exposing a large attack surface. One particular critical threat is wireless jamming, which allows malicious actors to perform denial of service attacks with minimal effort. In a classical jamming attack, the adversary transmits an interfering signal that overshadows the desired signal, preventing a victim receiver from correctly decoding it. Crucially, loss of connectivity impacts the functionality of wireless devices and can thus have potentially far-reaching consequences, such as in smart grids, smart transportation, and healthcare systems. Recent media reports underscore the real-world threat potential of jamming attacks, for example, criminals disabling smart home security systems, and preventing cars from locking.

This basic attack principle has previously been studied by a large body of research: For instance, the attacker can leverage various jamming waveforms, such as noise or replayed victim signals, and vary the attack timing, jamming constantly or only at certain times. As evident from the many existing attack strategies, wireless jamming has been incrementally refined and has become increasingly sophisticated. One particular example for this is the case of selective jamming attacks.

To illustrate a potential attack scenario, consider an adversary attempting to sabotage a complex automated manufacturing process. Distributed actuators might take orders from several previous processing stages that have to be executed in a timely fashion, risking manufacturing failure otherwise. Here, the adversary could use selective jamming to simulate local loss of connectivity on a single actuator but not the entire plant which would likely trigger some emergency response.

So far, the only means to realize such a selective jamming attack is via so-called reactive jamming where the attacker analyzes all wireless traffic in real-time to decide on-the-fly whether to send a jamming signal, relying on the existence of meaningful protocol-level information not protected by cryptographic primitives. In our manufacturing plant example, selective disruption of the actuator would require the attacker to receive and identify every packet directed to the recipient before sending a jamming signal. This restricts the attacker positioning rather close to the victim. Other downsides of this approach are that it can be mitigated by fully disguising packet destinations and the attack realization being rather complex and cumbersome.

In light of these aspects, we are interested in novel attack strategies resolving the aforementioned shortcomings. Clearly, the ideal solution would be to physically inject a proactive jamming signal directly and only into the victim device. But this is not possible due to the wireless nature of jamming and the inevitable broadcast behavior of radio signal

propagation to other, non-target devices. Thus, we aim to answer the following research question:

How can we physically target and jam one device while keeping others operational?

We solve this challenge by means of a reconfigurable intelligent surface (RIS) to devise the first selective jamming mechanism based on taming random wireless radio wave propagation effects. Using RIS-based environment-adaptive wireless channel control, allowing to maximize and minimize wireless signals on specific locations, the attacker gains spatial control over their wireless jamming signals. This opens the door to precise jamming signal delivery towards a target device, disrupting any legitimate signal reception, while leaving other, non-target devices, untouched. Other than reactive jamming, this is a true physical-layer selection mechanism, allowing realization independent of protocol-level information. Moreover, the attacker only initially needs to detect signals from considered devices, removing the need for any real-time monitoring and reaction to ongoing transmissions.

In this work, we experimentally evaluate RIS-based spatially selective jamming attacks against Wi-Fi communication, showing that it is possible to target one or multiple devices while keeping non-target devices operational. To accomplish this, we exploit that considered devices transmit signals, allowing the attacker to passively adapt to the scene. Apart from the attack's core mechanism, we study crucial real-world aspects such as the attack's robustness against environmental factors. We additionally verify the effectiveness of our attack in real-world wireless networks, where mechanisms that could counteract the attack are at play, for example, adaptive rate control of Wi-Fi networks. We show that RIS-based selective jamming even works despite extreme proximity of devices, for example, 5 mm, and investigate the underlying physical mechanisms. Finally, we perform comparison experiments with a directional antenna, showing the significance of our RIS-based approach.

In summary, our work makes the following key contributions:

• We propose the first true physical-layer selective targeting mechanism for wireless jamming, enabling environment-adaptive attacks in the spatial domain.

• We present an attack realization based on RIS, using passive eavesdropping to determine an appropriate RIS configuration which is the key to deliver jamming jamming signals towards targeted devices while avoiding non-target devices.

• We present a comprehensive experimental evaluation with commodity Wi-Fi devices, environmental changes, and an in-depth analysis of the physical properties of ourjamming attack.

One last note about these new RIS -- Reconfigurable Intelligent Surfaces. They write:

An RIS is an engineered surface to digitally control reflections of radio waves, enabling smart radio environments. It is worth noting that RISs are likely to become pervasive, as they hold the potential to complement future wireless networks such as 6G. Here, the propagation medium is considered as a degree of freedom to optimize wireless communication by redirecting radio waves in certain directions, for example, to improve signal coverage and eliminate dead zones, to enhance energy efficiency and data throughput, and building low-complexity base stations.

An RIS does not actively generate its own signals but passively reflects existing ambient signals. For this, it utilizes L identical unit-cell reflector elements arranged on a planar surface. Importantly, the reflection coefficient of each reflector is separately tunable to shift the reflection phase. Typically, an RIS is realized as a printed circuit board (PCB) with printed microstrip reflectors, enabling very low-cost implementation. To reduce complexity, many RISs use 1 bit control, for example, to select between two reflection phases 0◦ and 180◦, corresponding to the reflection coefficients +1 and −1. This allows the control circuitry to directly interface with digital logic signals from a microcontroller. The technology is still under development which is why RISs are currently not widely used in practice. At the time of writing, first implementations are being made commercially available, and field trials are being carried out.

And then, after many pages of very cool detail, their paper concludes:

In this paper, we investigated the merits of the RIS technology for active wireless jamming attacks. In particular, we have shown that the RIS enables precise physical-layer attack targeting in the spatial domain, enabling protocol level-agnostic selective jamming.

For this, the attacker first determines an RIS configuration by eavesdropping wireless traffic from the victim devices. Then, the attacker uses the RIS to reflect the environment's ambient radio signals with the effect of jamming the wireless communication of targeted devices while leaving other devices operational.

We have demonstrated the effectiveness of the attack under real-world conditions with extensive experimentation using commodity Wi-Fi devices and an open-source RIS. Notably, we found that it is possible to differentiate between devices that are located only millimeters apart from each other. Overall, our work underscores the threat of wireless jamming attacks and recognizes the adversarial potential of RISs to enhance the landscape of wireless physical-layer attacks.

Wow.

I know that our listeners enjoy being clued-in, even if with only the broad strokes I've been able to share here. Just knowing that such capability exists is interesting and potentially useful.

What this means, in practice, is that very low power undetectable targeted jamming of specific radios is now possible. It's low power because the device is not, itself, needing to emit any strong overwhelming radio signal. It's merely selectively inverting the reflected phase of what it receives across the elements of its two-dimensional surface. And this reflection property is also what makes this undetectable, again, because it's not emitting any flooding radio signal. It's also undetectable because the sum of these reflections can be focused onto the device exact antenna location so that even being half an inch away, no jamming effect would be detectable.

As I said earlier, I'd be very surprised if researchers at the NSA and CIA didn't already have their sleeves rolled up and taking a close look at what this means for our on-the-ground defensive and offensive operations.