Security Now! #936 - 08-22-23

When Heuristics Backfire

This week on Security Now!

Which Linux distro is selling itself to private equity capital and what could possibly go wrong? Will Android soon be talking to the sky? What's up with the trouble SanDisk and Western Digital are in over their SSDs? Are children still being tracked on YouTube's "made for kids" channels? Has cryptocurrency become any safer and what dangers are posed by the use of multi-party wallets? Is FIDO2 ready with post-quantum crypto? What's the latest on HTTPS by Default? And after looking at some feedback from our terrific listeners, we're going to examine the nature of heuristic programming algorithms with a case study in what can go wrong.

This is why they're sheep...

Security News

Thanks for the support

I wanted to take just a moment of everyone's time to thank the many listeners who let me know after last week's podcast how glad they were to learn that I would not be retiring from this weekly communication after podcast #999. It had become clear to me that the plan to quit for no particular reason after #999 would have felt very weird as that day was approaching. Now we know that's not going to happen.

OpenSUSE goes private

I'm not certain whether it will matter to anyone here, but I wanted to note that the currently public company SUSE, the company behind the OpenSUSE Linux distro, has announced its plans to delist itself from the Frankfurt stock exchange and allow itself to be purchased and taken over by a private equity firm. Their announcement read, in part: *"EQT Private Equity has underscored its commitment to supporting the company strategically and financially, and to co-operate closely with SUSE's CEO and his leadership team."* Dirk-Peter van Leeuwen, CEO of SUSE said: *"I believe in the strategic opportunity of taking the company private–it gives us the right setting to grow the business and deliver on our strategy with the new leadership team in place. EQT Private Equity's and SUSE's partnership in a private setting has been fruitful before and we are excited about the long-term potential of the company and our continued collaboration"*.

Right. The whole truth is, it's unclear what this means for the future of OpenSUSE. In general, and as we've witnessed a number of times in our own PC and security industry, private equity firms do not purchase tech companies out of their love of tech. They typically purchase them because they perceive pockets of unexploited profit potential and some means of squeezing money – in the short term – from the golden goose. This is typically done through aggressive layoffs, cost cutting, selling off the less profitable pieces, and in general squeezing the remaining life – and the future – out of their new acquisition. Users who have made their own investments in the future of such enterprises should be forewarned that, if nothing else, change is coming. https://www.suse.com/news/EQT-announces-voluntary-public-purchase-offer-and-intention-to-delist-SUSE/

Android to get satellite comms

If imitation is the greatest form of flattery, it appears that Google is enamored of the satellite communications capability that Apple added to iOS 14. It appears that Google is testing a similar new Android feature that will allow its users to send SOS messages via a satellite connection in the case of an emergency. The nascent feature was spotted in the code for Android 14 which is slated for release within days. The feature is unannounced and should not be expected to be going live anytime soon. According to Android OS expert Mishaal Rahman, the feature is not part of the main Android 14 OS and is tagged as demo code. And, of course, it's going to need accompanying radio hardware. But Android users can probably assume that a feature similar to Apple's will eventually arrive.

SanDisk and Western Digital in hot water

I should mention, in case any of our listeners might be impacted by this news, that SanDisk (my own preferred manufacturer of solid state mass storage) and Western Digital are both currently in the dog house over what is alleged to be their willing and knowing sale of defective SSD products. That's a disturbing allegation. Here's the start of what ArsTechnica said about this:

> *Amid ongoing pressure to address claims that its SanDisk Extreme SSDs are still erasing data and becoming unmountable despite a firmware fix, Western Digital is facing a lawsuit over its storage drives. A lawsuit filed on Wednesday accuses the company of knowingly selling defective SSDs.*
>
> *Western Digital brand SanDisk series of Extreme V2 and Extreme Pro V2 portable SSDs are often recommended by tech review sites. If you've considered a portable drive, it's likely you've come across the popular series in your search.*
>
> *However, numerous owners of the drives, including Ars Technica's own Lee Hutchinson, encountered a problem where the drives seemingly erased data and became unreadable. Lee saw two drives fill approximately halfway before showing read and write errors. Disconnecting and reconnecting showed the drive was unformatted and empty. Wiping and formatting didn't resolve things.*
>
> *Complaints about the drives littered SanDisk's forums and Reddit* [Ars then provides 4 example links] *for at least four months before Western Digital released a firmware fix in late May. The page for the update claims products currently shipping are not affected. But the company never noted customers' lost data claims.*
>
> *It did, however, name the affected drives:*
>
> *SanDisk Extreme Portable 4TB (SDSSDE61-4T00)*
> *SanDisk Extreme Pro Portable 4TB (SDSSDE81-4T00)*
> *SanDisk Extreme Pro Portable 2TB (SDSSDE81-2T00)*
> *SanDisk Extreme Pro Portable 1TB (SDSSDE81-1T00)*
> *Western Digital My Passport 4TB (WDBAGF0040BGY)*
>
> *Subsequent reports from The Verge, which received a replacement SSD, and some Reddit users, though, claimed the drives were still broken. Western Digital didn't answer requests for comment about newfound grievances.*

So, for what it's worth, I just wanted to inform our listeners that this has been and apparently still is a problem. And if I were storing important data on any of those named devices I'd be nervous.

Hearing that the drives started having trouble when they became around half full was interesting to me. As an engineer I've been astounded and shaking my head when I learn the lengths to which today's SSDs are going to achieve the storage densities they offer. For example, get a load of this: Remember first that a NAND-style SSD storage cell is really just a capacitor. It's an

itty-bitty (that's a valid engineering term) spec of metal sitting on top of an insulating coating. The act of writing to that cell involves cranking up the voltage to deliberately breakdown that layer of insulation to inject and strand electrons on that itty-bitty island of metal. Once there, their presence can be sensed thanks to the magic of field-effect transistors. But the point is, the number of electrons which have been stranded on that electrostatic island is an inherently analog quantity.

The designers of the earliest FLASH memory, which regarded the fact that all of this worked at all to be a miracle, were quite content to only fully fill or empty the islands to store 0's and 1's. But then management came along and demanded more density. The engineers explained that they had already made the electrostatic islands as itty-bitty as possible. But then one of them made the mistake of suggesting during a brainstorming meeting that they might be able to store two bits of data on each island by storing varying amounts of electrons, thus doubling the effective storage density of the same number of itty-bitty memory cells.

Instead of storing a boring old 0 or 1, it would be possible to store 0, 1/3rd, 2/3rds, or 1. Since this gives us four separate storage levels, that's two bits of data. Whereas the original NAND memory cells were known as SLC – for single level cell – this next generation were MLC – for multi-level cell.

And, of course, once you've opened Pandora's Box of storing what are essentially analog levels of electrostatic charge in cells, why stop at two bits? How about three or four?

So, with this bit of background, here's what I learned that made me shake my head: The SSD manufacturers are aware that storing fewer bits per cell is better. It's faster to read and write and more reliable. So, believe it or not, today's most advanced SSDs start out by storing fewer bits per cell **because** it's faster and more reliable. It makes their product benchmark faster. But as their user continues to fill them up with data, the SSD begins to run out of storage at that lower, faster and safer density. So at some point of "fullness" it needs to switch over to storing more bits per cell in order to actually deliver its promised and rated full storage capacity. To do this, it starts reading and rewriting existing data which was stored at the lower bit density into higher bit density. Essentially, it is on-the-fly cramming more bits into fewer cells because, as it turns out, its owner actually wants to use the storage capacity that they purchased.

The extra complexity that had to be built into the controller to track and accomplish all of this is somewhat mind boggling, but it's obviously worth it to its product's designers. The flakiness inherent in how far SSD storage density has been pushed is one of the reasons I became certain that a continuing investment in SpinRite would be warranted.

Only SanDisk's engineers know why, when their drives were storing about half of their rated capacity they suddenly developed a problem that crashed the entire drive. It could be that the drive had actually filled itself with data stored at half-density and now needed to get serious about doubling-up on density; and that some flaw in the drive's firmware was then triggered. We'll likely never know. This brings to mind something else that happened and was discovered last week by one of SpinRite's testers on one of their thumb drives. But I've already taken up a lot of time on this, so I'll share that interesting story next week.

You're asking for it!
I ran across a blurb of news that read: "YouTube children's privacy: An Adalytics report found that advertisers are still tracking viewers of videos made for kids, despite a 2019 promise from YouTube to stop delivering personalized ads on these types of videos. Senators are now seeking a formal inquiry into the company for breaching the US' COPPA laws."

COPPA is the Children's Online Privacy Protection Act. I was curious, so I followed the Adalytics link. Adalytics appears to be a scrupulously neutral Adtech industry watchdog. Their report of long, but I'll share just the top few takeaways which will give everyone the gist:

1. YouTube's CEO said in 2019 that the platform would "limit data collection and use on videos made for kids only to what is needed to support the operation of the service;" however, YouTube appears to be setting or transmitting "advertising" cookies and identifiers on the devices of viewers who are watching "made for kids" videos as of July 2023.

2. YouTube's CEO said in 2019 that the platform would stop serving personalized ads on "made for kids" content. However, demographically and behaviorally personalized ad campaigns appear to have ads being served on "made for kids" YouTube channels as of July 2023.

3. YouTube is serving ads from many ('adult') Fortune 500 advertisers and major media agencies on YouTube channels that are labeled as "made for kids." These include major brands such as Mars, Procter & Gamble, Ford, Colgate-Palmolive, Samsung, and many others.

a.) In some (adult) brands' personalized ad campaigns, the top YouTube channels by clicks or clickthrough rate are popular "made for kids" YouTube channels such as "ChuChu TV Nursery Rhymes & Kids Songs", "CoComelon Nursery Rhymes & Kids Songs", or "Kids Diana Show."

4. The viewers of "made for kids" YouTube videos appear to be clicking on ads... and brands' websites – such as Michigan State Police, Disney, BMW, Hyundai, and Verizon - are harvesting and sharing meta-data on those viewers with dozens of data brokers upon click through. This raises the possibility that brands have "data poisoned" their first party datasets with data derived from thousands of viewers of "made for kids" videos.

5. Dozens of major ad tech and data broker companies are receiving data from viewers of "made for kids" YouTube videos who clicked on an ad – these include several companies who paid penalties for COPPA related enforcements, such as Amazon, Facebook, Microsoft, and OpenX. Foreign-owned companies such as TikTok are also receiving meta-data from viewers of "made for kids" content from advertisers' websites.

And it goes on and on. This is clearly not that the COPPA Act had in mind, so it's hardly surprising that the U.S. Senate, which has been all worked up over TikTok, is wondering what's going on with this popular U.S.-owned property. In the show notes, I titled this story "You asked for it" since that's clearly what Google is doing. This entire tracking and data-brokering industry needs a flushing... and at this rate that might not be too far away.

Whoopsie!
Since we were just talking about the danger of losing access to an eMail forwarding service, I got a kick out of this bit of news: Apparently the operators of the **8Base** ransomware operation lost all of the data they had previously stolen from their victims when the **AnonFiles** cloud hosting file storage service closed down without notice last week. Whoopsie!

Where the money is
Meanwhile, the blockchain security firm CertiK who often provides interesting tidbits says that a group of Canadian scammers are responsible for stealing millions of dollars during the past few years. The group's members operate by hacking the Discord servers used by cryptocurrency communities and posting phishing links to hijack their users' wallets. CertiK claims to have identified the real-world identity of one of the group's members. Going by the pseudonym of "Faint", the individual is believed to have stolen more than $1 million worth of assets since late 2022. "Faint" is the group's second publicly-identified member after another blockchain investigator exposed a hacker named "Soup" last month.

I just tossed that one in to remind everyone that nothing much appears to have changed in the cryptosphere. It remains a wildly immature and wildly insecure mess.

The TSSHOCK vulnerability
Here's another example of this fundamental immaturity: Major crypto-industry players jumped onto unproven and not-fully tested algorithms due to their promise. (And isn't that the story of the entire cryptocurrency phenomenon?) Earlier this month, during BlackHat, researchers from Verichain disclosed three vulnerabilities in a cryptographic protocol called TSS which is used to create multi-party crypto-wallets, known as MPCs. Here's how Binance, one of the prominent users of this new TSS algorithm describes it on their site:

> *Threshold Signature Scheme (TSS) is a cryptographic primitive for distributed key generation and signing. The use of TSS in blockchain clients is a new paradigm that can provide numerous benefits, especially in terms of security.* **[Whoops!]** *In the broader sense, TSS can influence the design of key management systems (such as crypto wallets) and lead the way for native support in DeFi use cases. Having said that, TSS is still a new technology, so the risks and limitations should also be considered.*

That last part they got right; but that didn't prevent them from deploying it. The Verichain researchers named their attacks on TSS "TSSHOCK". TSSHOCK exploits vulnerabilities in some of these MPC – multi-party crypto-wallets – through their implementation of the threshold elliptic curve digital signature algorithm (ECDSA). The exploit of TSSHOCK can allow threat actors to steal cryptocurrency from individual users or major institutions while leaving no trace of the attack on the client side. And here's the good part: Major companies like Binance, ZenGo, Multichain, THORChain, and ING Bank use exactly these vulnerable threshold ECDSA software implementations. And on top of all that, TSSHOCK is the second major multi-party crypto-wallet vulnerability disclosed this month, the first being BitForge.

BitForge

For the sake of painting a full picture of just how rocky things still are in the crypto industry, I should also touch on this earlier discovery which was made by a group known as Fireblocks. Two weeks ago, on August 9th, Fireblocks posted their discovery under the title: "Fireblocks Researchers Uncover Vulnerabilities Impacting Dozens of Major Wallet Providers." And they explained:

> *Today, the Fireblocks Cryptography Research Team announced the discovery of multiple 0-day vulnerabilities in some of the most used cryptographic multi-party computation (MPC) protocols, including GG-18, GG-20, and implementations of Lindell 17. If left unremediated, the exposures would allow attackers and malicious insiders to drain funds from the wallets of millions of retail and institutional customers in seconds, with no knowledge to the user or vendor. The series of vulnerabilities, dubbed BitForge, impact popular wallet providers like Coinbase WaaS (I guess that's Wallet as a Service), Zengo, and Binance. Following the industry-standard 90-day responsible disclosure process, Coinbase WaaS and Zengo have since fixed and resolved the identified issues. In addition, the academic papers which had the details of their flaws redacted have been revised.*
>
> *The Fireblocks Cryptography Research Team findings were presented during the Black Hat USA conference on Wednesday, Aug 9, and will be shared at Defcon on Thursday, Aug 10.*
>
> *Pavel Berengoltz, Co-founder & Chief Technology Officer at Fireblocks said: "As decentralized finance and Web3 continue to gain popularity, the need for secure wallet and key management providers is evident. While we are encouraged to see that MPC is now ubiquitous within the digital asset industry, it is evident from our findings — and our subsequent disclosure process — that not all MPC developers and teams are created equal. Companies leveraging Web3 technology should work closely with security experts with the know-how and resources to stay ahead of and mitigate vulnerabilities. Maintaining and updating core infrastructure technologies, like Web3 wallets, is crucial in preventing thefts and attacks, which amounted to* **[ready for this]** *nearly* ***$500 million*** *in just the first half of 2023."*

"Wallet as a Service" – wow. From the beginning of this cryptocurrency odyssey our one constant piece of advice has been to keep your wallet safely offline. Might you miss having a convenience feature or two? Yes. Good. Better that than missing all of your money!

Anyway, enough said there. If you want to play with crypto, just please be careful and heed the age old investment advice to never gamble more than you're able to lose.

A Quantum resilient security key

Anyone want a "quantum resilient" FIDO2 security key? It may be of more interest once it's possible to use passkeys widely. But in any event, Google has developed the first-ever version of a FIDO2 security key that includes protections against quantum computing attacks. The implementation comes from a new version of OpenSK (as in Open Security Key) which is an open-source project on Github that provides firmware for security keys. Google says this new OpenSK firmware version uses a novel ECC/Dilithium hybrid signature schema its engineers developed with academics from ETH Zurich. The project doesn't yet appear to be production ready. It's definitely still experimental and not ready for prime time. But it appears that when quantum computers eventually happen, we'll be ready – even if there still aren't many sites

supporting Passkeys!

Removed Chrome extensions notifications

Chrome 116, which is today's current release, contains a not-enabled-by-default feature which is scheduled to be enabled by default in Chrome 117 at the start of September. The feature is called "safety check extensions" which will allow Google to display notifications to Chrome users when one or more browser extensions that they currently have installed are removed for some reason from the official Web Store. The notifications will be shown when an extension is marked as malware, the extension is removed for ToS violations, or when the extension is unpublished by its developer. That seems like an uncontroversial good thing, though it's a bit surprising that Chrome doesn't already do that.

HTTPS by default?

Even with TLS certificates now being free, we're still not all onboard the HTTPS train. So last Wednesday, in a posting titled "Towards HTTPS by default" Google announced their next move:

> *For the past several years, more than 90% of Chrome users' navigations have been to HTTPS sites, across all major platforms. Thankfully, that means that most traffic is encrypted and authenticated, and thus safe from network attackers. However, a stubborn **5-10%** of traffic has remained on HTTP, allowing attackers to eavesdrop on or change that data.*

Okay. Wait. It's not "stubborn". It's just ignoring you, Google. It doesn't care! Imagine that. As we know, this is certainly traffic that really doesn't need any security of any kind and is 100% completely happy being unauthenticated and out in the open. But it annoys Google nevertheless.

I have an example in the show notes of exactly such a site. I refer to it constantly while working on SpinRite since it contains a comprehensive list of every PC interrupt from 00 to FF, all of their sub-functions and arguments. It's wonderful and I'm very thankful for it. I've copied the entire site against the inevitable day when it finally disappears: http://www.ctyme.com/intr/int.htm And yes, it's proudly HTTP and I'm pretty sure that's never going to change. It also makes my own website look quite modern by comparison – it even has the old mailbox icon for eMail. Anyway, Google is still annoyed, so here's what they're going to do:

> *Chrome shows a warning in the address bar when a connection to a site is not secure, but we believe this is insufficient: not only do many people not notice that warning, but by the time someone notices the warning, **the damage may already have been done!**.* [Oh the horror! (What damage?) From looking up some interrupts in a table? That's right, some dastardly Russian could subvert SpinRite!]
>
> *We believe that the web should be secure by default.* [That's right! No insecure traffic allowed! Boy, you'd think they were selling certificates. I think they're getting a little far over their skis here. After all, they're just a web browser. Somebody had too much time on their hands.] ***HTTPS-First Mode** (that's the new thing) lets Chrome deliver on exactly that promise* [no insecure interrupt tables!]*, by getting explicit permission from you before connecting to a site insecurely.* [Oh, great. Thank goodness I'm over here on Firefox.] *Our goal is to eventually*

enable this mode for everyone by default. While the web isn't quite ready to universally enable HTTPS-First Mode today, we're announcing several important stepping stones towards that goal.

Chrome will automatically upgrade all http:// navigations to https://, even when you click on a link that explicitly declares http://. This works very similarly to HSTS upgrading, but Chrome will detect when these upgrades fail (for example, due to a site providing an invalid certificate or returning a HTTP 404), and will automatically fallback to http://. This change ensures that Chrome only ever uses insecure HTTP when HTTPS truly isn't available, and not because you clicked on an out-of-date insecure link. We're currently experimenting with this change in Chrome version 115, working to standardize the behavior across the web, and plan to roll out the feature to everyone soon. While this change cannot protect against active network attackers [Right. We must protect against the pollution of the PC's interrupt table!]*, it's a stepping stone towards HTTPS-First mode for everyone and protects more traffic from passive network eavesdroppers.*

Okay. So overall, this is a good thing. It **would** be annoying to need to push past another warning screen before being allowed to access an HTTP site. That does seem like overdoing it. But certainly trying first to access a URL over port 443, bring up a TLS handshake and see whether that HTTP:// URL is available via HTTPS:// makes sense.

It is worth noting, though, that technically HTTP:// URL's and HTTPS:// URL's are actually referring to different resources. Nothing anywhere ever says that the URLs of those differing protocols always or ever needs to or should refer to the same web resources. It's true that they generally do. Most web servers serve the same content regardless of whether their visitors come in via HTTP or HTTPS. And these days, most attempts to come in via HTTP are immediately redirected to the same URL over HTTPS. But that doesn't necessarily need to be so. It's just the way things have typically evolved over time. So I can see why Google is being cautious. They clearly want to kill HTTP entirely and I'll bet they even want to remove its support from Chrome. But it doesn't look like it's going to go quietly.

WinRAR 6.23 final released

I've long been a fan and a registered user of WinRAR. So I wanted to note that a recent update to v6.23 earlier this month closed a couple of maliciously exploitable holes that attackers could use in a targeted attack to run their own code on their victim's system.

Okay... let's close some loops:

Closing the Loop

Iam-py-test / @iam_py_test

Hello, Hope your weekend is going well. Relating to this weeks episode, I thought you might find it interesting to know uBlock Origin disables Topics by default: https://github.com/uBlockOrigin/uBlock-issues/discussions/2714#discussioncomment-6396384
It does this by modifying the Permissions Policy header. Thanks for the podcast and have a nice day.

I suppose we should have expected this from Gorhill. In the Github dialog on this, Gorhill quoted a comment from someone else, which read: *"@stephenhawk8054 Gave more thoughts to this and I think we should add it to uBlock filters -- Privacy list. I don't think a user interface component for this is the way to go given that so far it's an API supported only by one browser, and that the API is to serve advertisers in the first place."* As we know, since Topics is a new component of the underlying Chromium browser core, all of the many Chromium-based web browsers will at least have the opportunity to support it easily – and initially, some of those may choose to block it. This sort of major change can be expected to take time and will require a great deal of re-education, which we, at least, began on this podcast last week.

Everyone here enjoys at least knowing where the bleeding edge is – and we always find it. So some patience will be needed. The EFF will likely never come around on the idea of helping websites to monetize their visitors by providing advertisers with any information about who's viewing their pages. And maybe they're right. It might be that all of this profiling which grew out of tracking which was itself an unintended side effect of cookies has always been something that data brokers have been getting away with only until slow moving government regulations finally catch up. Perhaps absolute anonymity is what we'll eventually get thanks to legal frameworks rather than technological frameworks. No matter what happens, we'll have fun following it here.

And if Topics succeeds with universal cross-browser adoption and industry endorsement while Gorhill's uBlock Origin continues to block it, then we'll ask Gorhill for an easy to use exception switch to turn it back on, or figure out which blocking rules to delete from its privacy list.

The Real Veran Dontic / @RealVeranDontic

Hi, Mr. Gibson. I love Security Now. I noticed a slight logical flaw on SN935 regarding the GPL. The GPL only requires that you make available the source code of your modified GPL code with the distribution of that code. Russian citizens could safely honor the GPL without actively contributing to open source.

First of all, it sounds like he's been listening to Ant with this "Mr. Gibson." I appreciate the gesture of respect, but please everyone, we've been together for 18 years. I'm just "Steve." And I do appreciate Veran's clarification about the GPL after I fumbled my mention of it. But even so, assuming that Russia invests in modifying Linux to suit their own purposes and thus acquires an inherent feeling of propriety over "their" result. Who would imagine that Vladimir Putin's minions will have any interest in providing their citizens with the source code – any more than Microsoft

does with Windows – or that they would make it available to anyone, including the Linux Project, in any form or fashion? What benefit would Russia accrue from doing so? I'd be willing to bet that we're going to see the Linux source forked and never shared again.

Emma Sax / @emmahsax

*I just finished listening to your latest podcast, and I've been doing some thinking on your concerns about an email forwarding service, like Fastmail, just choosing to shut down. I've come to the conclusion that that concern is no different than Gmail or Outlook shutting down their servers, which millions of people rely on. At any point, any common email providers could just choose to not host email anymore, and tons of people would be forced to change their emails all over. The only way to get around it is by running your own email server, which you mention. And *most* people aren't going to go that far. So for most people, we might as well just accept that we're relying on other peoples' servers for things. And so just do your research to make sure you're choosing one that seems the most reliable to you (as in owned by a big company, versus a single person, etc).*

I think Emma is exactly right. I think I'm sensitive to the fact that we do see services come and go, even from major providers. Mozilla once offered a terrific file transfer service which they shutdown because it was being abused to transfer malware. And we've seen several companies stop offering ephemeral credit card numbers which was a useful service. The difference with those services is that they don't incur the same level of persistent dependence that an eMail forwarding service does when it's used to anchor online account recovery. So, Emma's right. I think that the optimal thing to do is to choose your eMail forwarding provider with your eyes open and with an awareness of the importance of any such service's continuing support and existence. And if you ever receive any notice of any pending service discontinuation don't wait to migrate your existing forwarded accounts to some other provider.

AMSather / @ASather2775

Hey Steve topics sounds great, but what's to stop site operators from storing the topics supplied to them (via fingerprinting or other means of tracking) and aggregating it to sell to advertisers? Love the show. Longtime listener and SpinRite6 owner. So happy you're not stopping after 999

This is one of the trickier bits of the Topics API. And I hope its subtlety keep it from being appreciated. Remember that brain twisting part about which topics a requestor could receive from a browser? One of the things to appreciate about the browser's role in this is just how much of the burden for enforcing the user's privacy is placed on the browser. It needs to keep track of all kinds of information. But, fortunately, only for a moving 4-week window.

In order to obtain those three interest Topics from a user's browser when the user is visiting a website, the requestor – who would typically be an advertiser serving ads across the Internet – must have previously queried that user's browser at some site which the browser associates with each of the topics that would be returned to the requesting advertiser.

Let's go with an example. Three topics will be chosen, one for each of the past three weeks, based upon the domain the user is visiting. Say that one of the three selected topics is "Fish." This would happen if the user's use of their web browser during the previous three weeks has temporarily taught their browser, because of the web sites they have chosen to navigate to, that they currently have an interest in fish. So, "Fish" would be among their current top interests.

But at this time they are not at a website that their web browser associates with "Fish". They're at a site that their web browser associates with gaming. But nonetheless, an advertiser on that gaming site is asking for the three chosen topics of interest to this user now. And here's the crux: "Fish" will only be returned as one of the three topics to that advertiser if sometime during the preceding three weeks that same advertiser queried for the user's three topics at a site which the user's browser associates with "Fish".

In other words, in order to be told that this user at this gaming site has an interest in fish, that same advertiser needs to have recently previously encountered this user's browser at a website that the browser associates with fish. Unless that's true, "Fish" will be eliminated from the three topics that will be returned to the user and nothing will be substituted in its place.

Okay. now our listener asked: *"what's to stop website operators from storing the topics supplied to them and aggregating them to sell to advertisers?"* And that's part of the subtle beauty of this system. Ask yourself what topics a browser will return to any static website? Unlike advertisers, websites don't encounter their visitor's browsers at other websites – only 3rd-party advertisers have that sort of cross-website reach. Websites have a 1st-party relationship with their visitors. So they never see them anywhere else. That means that the tricky Topics filter will only ever return the same topics to a website that the browser associates **with that website**. This means that websites are unable to learn **anything** they didn't already know about their visitors.

Note that this is completely different from FLoC which blabbed with that cryptic hashed token to every site someone visited. No so with Topics.

Robert Gauld / @RobertGauld

@SGgrc in SN935, I think consecutive characters refers to sequences as in ABC, def, 567 etc.

Yes. Thanks to Robert and many others who said: "Uhhhh, Steve... I think that what they were trying to say was "sequential" characters but they wrote "consecutive" characters. Looking at it again, that the only thing that makes sense.

Rob Mitchell / @TheBTCGame

Ya know what would be great? If you posted your picture of the week for each episode here on Twitter. You could include a link to each episode, so you'd also be doing some promotion too. I rarely take the time to find the photo each week, but I'd surely find it here!

When Heuristics Backfire

Wikipedia defines the term "heuristic" as:

> *A heuristic (/hjʊˈrɪstɪk/; from Ancient Greek εὑρίσκω (heurískō) 'to find, discover'), or heuristic technique, is any approach to problem solving or self-discovery that employs a practical method that is not guaranteed to be optimal, perfect, or rational, but is nevertheless sufficient for reaching an immediate, short-term goal or approximation. Where finding an optimal solution is impossible or impractical, heuristic methods can be used to speed up the process of finding a satisfactory solution. Heuristics can be mental shortcuts that ease the cognitive load of making a decision. Examples that employ heuristics include using trial and error, a rule of thumb or an educated guess.*

Before we take some lessons from the implementation of a heuristic that has been causing Microsoft enterprise customers years of mysterious pain, I want to note that I'm a big fan of heuristic methods. I often use heuristics in my own code when I'm dealing with uncertainties. An example from the nearly-finished SpinRite 6.1 code comes to mind:

Modern operating systems all incorporate the concept of device drivers because the operating system needs to be informed how to talk to the wide range of peripherals it might encounter. For all SpinRites from v1.0 through 6.0 this was never an issue because SpinRite was able to access the system's mass storage hardware through the BIOS which functioned as an abstraction layer. Essentially, it contained permanent device drivers so that it knew how to talk to the hardware on its motherboard. The problem was that the BIOS APIs were designed 40 years ago. And mass storage devices have evolved not only in their capacity but also in the rich metadata that they're able to offer to anyone who knows how to ask. The BIOS never did.

So, to pull off what SpinRite 6.1 needed to do, it could no longer use the BIOS to serve as its intermediary. It needed to talk to the motherboard hardware directly in order to gain direct access to all of the modern hardware capabilities. But unlike all modern operating systems which are able to use a rich collection of modular device drivers to talk to the hardware, SpinRite needed to have what was essentially a single universal built-in driver that could figure out, for itself and on its own, how to talk to any hardware that it might encounter. And that's where heuristics comes in.

While SpinRite is starting up, it spends time literally getting to know the hardware it's running on by performing a series of experiments – in a classic heuristic process – to work out and learn exactly how the underlying hardware operates. It tries things and learns things and keeps a record of what it has learned about every mass storage adapter and device in the system so that it knows how to interface with each one. There was an appreciable amount of time during these past three years where I wasn't 100% certain that I was going to be able to successfully weave a path through all of the hardware owned by me and by SpinRite's 759 individual testers. But a path was found and it's been quite a while since any hardware has stumped SpinRite. It doesn't matter how old or new or from which manufacturer. SpinRite v6.1 now has what is essentially a universal smart driver – driven by a heuristic learning process – that's able to interface with any IDE, ATA or AHCI mass storage hardware that's ever been built. And one non-obvious feature of the inherent flexibility of this approach is that the benefit may not only be retrospective. It's

likely also prospective – giving SpinRite the ability to figure out things that not only came before it, but that may also arrive after it.

So that's an example of the application of heuristics. One way to think of it is as code that figures things out, often working from an initial starting place of uncertainty. The trouble is, being somewhat "rule of thumb" and "inexact", anyone deploying heuristic approaches needs to build- in safeguards against their code drawing the wrong conclusions, and this appears to be where a heuristic approach incorporated into Windows Servers by Microsoft back in 2016 has fallen down, backfired, and caused unappreciated problems for the past seven years.

Last week, a Norwegian data center engineer named "Simen" (spelled Simen) reached out to me via Twitter DM and explained that people who listen to this podcast said that I'd probably be interested in what he had discovered. Given that it became today's topic, it's clear that our podcast listeners have come to know me pretty well. The best way to introduce the seven year old problem that Simen uncovered would be to share ArsTechnica's coverage of it, which Simen pointed me to in his direct message:

Last Wednesday's piece was titled: *"Windows feature that resets system clocks based on random data is wreaking havoc"* and the subtitle was *"Windows **Secure Time Seeding** resets clocks months or years off the correct time."* Ars wrote:

> *A few months ago, an engineer in a data center in Norway encountered some perplexing errors that caused a Windows server to suddenly reset its system clock to 55 days in the future. The engineer relied on the server to maintain a routing table that tracked cell phone numbers in real time as they moved from one carrier to the other. A jump of eight weeks had dire consequences because it caused numbers that had yet to be transferred to be listed as having already been moved and numbers that had already been transferred to be reported as pending.*
>
> *The engineer, who asked to be identified only by his first name, Simen, wrote in an email: "With these updated routing tables, a lot of people were unable to make calls, as we didn't have a correct state! We would route incoming and outgoing calls to the wrong operators! This meant, for example, children could not reach their parents and vice versa."*
>
> *Simen had experienced a similar error last August when a machine running Windows Server 2019 reset its clock to January 2023 and then changed it back a short time later. Troubleshooting the cause of that mysterious reset was hampered because the engineers didn't discover it until after event logs had been purged. The newer jump of 55 days, on a machine running Windows Server 2016, prompted him to once again search for a cause, and this time, he found it.*
>
> *The culprit was a little-known feature in Windows known as **Secure Time Seeding**. Microsoft introduced the time-keeping feature in 2016 as a way to ensure that system clocks were accurate. Windows systems with clocks set to the wrong time can cause disastrous errors when they can't properly parse timestamps in digital certificates or they execute jobs too early, too late, or out of the prescribed order. Secure Time Seeding, Microsoft said, was a hedge against failures in the battery-powered onboard devices designed to keep accurate time even when the machine is powered down.*

Microsoft engineers wrote: "You may ask—why doesn't the device ask the nearest time server for the current time over the network? Since the server is not in a state to communicate securely over the network, it cannot obtain time securely over the network as well, unless you choose to ignore network security or at least punch some holes into it by making exceptions."

To avoid making security exceptions, Secure Time Seeding sets the time based on data inside an SSL handshake the machine makes with remote servers.

When I read that my first thought was "What?!?!" Ars explains some things we know but adds some new bits. They wrote:

These handshakes occur whenever two devices connect using the Secure Sockets Layer protocol, the mechanism that provides encrypted HTTPS sessions (it is also known as Transport Layer Security). Because Secure Time Seeding (STS) used SSL certificates Windows already stored locally, it could ensure that the machine was securely connected to the remote server. The mechanism, Microsoft engineers wrote, "helped us to break the cyclical dependency between client system time and security keys, including SSL certificates."

Simen wasn't the only person encountering wild and spontaneous fluctuations in Windows system clocks used in mission-critical environments. Sometime last year, a separate engineer named Ken began seeing similar time drifts. They were limited to two or three servers and occurred every few months. Sometimes, the clock times jumped by a matter of weeks. Other times, the times changed to as late as the year 2159.

Okay. Now let's just pause here for a moment. This is a perfect example of a very poorly designed and thought out heuristic. We don't yet understand in detail what's going on. We don't know exactly what Windows is doing to obtain its time of day and date information. But we don't need to, yet. The very fact that it's even **possible** for a server last year in 2022 to believe that it's now 2159 – 137 years in the future – conclusively demonstrates that the designer of this system left out a crucial concept which is very important in heuristic systems. My own name for them is "Sanity Checks" and our listeners will have heard me refer to them from time to time because they're integral to creating robust systems. As its name suggests, a "sanity check" is a "reasonability filter" that's applied once an answer is obtained from a heuristic system. It prevents believing ridiculous or impossible things.

So, what sanity checks might be applied here? Well... the Microsoft engineers were bragging about how they could rely upon the machine's locally stored TLS certificates to allow them to obtain a secure connection. So it would be reasonable to expect those certificates to have valid expiration dates, right? But 137 years in the future all of those certificates will have expired. That would appear to fail the "reasonability" test. What operating Windows server would have all of its certificates expired by well more than 100 years?

Or how about asking the local Windows Update service for the timestamp on the most recently received monthly update? Is it reasonable to imagine that the machine hasn't had a security or feature update in 137 years? Given Microsoft's track record for bugs? No, it's not.

ArsTechnica continues with a quote from Ken, who witnessed that:

Ken wrote in an email: "It has exponentially grown to be more and more servers that are affected by this. In total, we have around 20 servers (VMs) that have experienced this, out of 5,000. So it's not a huge amount, but it is considerable, especially considering the damage this does. It usually happens to database servers. When a database server jumps in time, it wreaks havoc, and the backup won't run, either, as long as the server has such a huge offset in time. For our customers, this is crucial."

Gee... seems like the backup system is smart enough to say "Uhhh... something's wrong here!" but Windows itself is blissfully ignorant.

Simen and Ken, who both asked to be identified only by their first names because they weren't authorized by their employers to speak on the record, soon found that engineers and administrators had been reporting the same time resets ***since 2016****.*

In 2017, for instance, a Reddit user in a sysadmin forum reported that some Windows 10 machines the user administered for a university were reporting inaccurate times, in some cases by as many as 31 hours in the past.

Here again, another missed opportunity for a sanity check. In our current reality, time stubbornly only moves forward. It turns out that this really simplifies things. But that means that discovering that many files in its local file system are suddenly timestamped 31 hours in the future might give any well-designed heuristic algorithm pause. In any event, Ars write:

The Reddit user eventually discovered that the time changes were correlated to a Windows registry key in

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits.

Additional investigation showed that the time changes were also linked to errors that reported valid SSL certificates used by the university website were invalid when some people tried to access it.

Gee, imagine that.

The admin reached the following conclusion: Windows 10 (and 11) have a feature called Secure Time which is ***on*** *by default. It correlates time stamp metadata from SSL packets and compares them with time the machine's local time. It processes these various times by means of "black magic" and sets the system clock accordingly. Unfortunately, this feature has the tendency to "flip out" and set the system time to a random time in the past or the future. The flip out MIGHT be caused by issues with SSL traffic.*

Other examples of people reporting the same behavior date back to 2016, shortly after the rollout of STS and many more recent reports of harmful STS-induced time changes have been reported.

One Reddit user wrote. "We've run into a show-stopping issue where time on a bunch of production systems jumped forward 17 hours. If you've been in the game more than a week, you know the havoc this can cause."

Okay. So what's going on? To determine the current time, STS pulls two pieces of metadata contained in the SSL handshake:

ServerUnixTime, which is a date and time obtained from the number of seconds that have elapsed since 00:00:00 UTC on January 1, 1970.

The other is cryptographically signed data obtained from the remote server's SSL certificate showing whether it has been revoked under a mechanism we've discussed here at length: OCSP – the Online Certificate Status Protocol.

Here's how Ars explains what **it** was told by Microsoft's engineers:

> *Microsoft engineers said they used the ServerUnixTime data "assuming it is somewhat accurate" but went on to acknowledge in the same sentence that it "can also be incorrect." To prevent STS from resetting system clocks based on data provided by a single out-of-sync remote server, STS makes randomly interspersed SSL connections to multiple servers to arrive at a reliable range for the current time. The mechanism then merges the ServerUnixTime with the OCSP validity period to produce the smallest possible time range and assigns it a confidence score. When the score reaches a sufficiently high threshold, Windows classifies the data as an STSHC, short for Secure Time Seed of High Confidence. The STSHC is then used to monitor system clocks for "gross errors" and correct them.*

What you've just heard is a perfect textbook example of a heuristic algorithm.

Quote: "*The mechanism then merges the ServerUnixTime with the OCSP validity period to produce the smallest possible time range and assigns it a confidence score. When the score reaches a sufficiently high threshold, Windows classifies the data as an STSHC, short for Secure Time Seed of High Confidence.*"

It doesn't get any more heuristic than that! Unfortunately, it is also apparently somehow prone to misfiring badly to become "highly confident" about very wrong times. Ars says:

> *Despite the checks and balances built into STS to ensure it provides accurate time estimates, the time jumps indicate the feature sometimes makes wild guesses that are off by days, weeks, months, or even years.*
>
> *Ken wrote in his email: "At this point, we are not completely sure why secure time seeding is doing this. Being so seemingly random, it's difficult to [understand]. Microsoft hasn't really been helpful in trying to track this, either. I've sent over logs and information, but they haven't really followed this up. They seem more interested in closing the case."*
>
> *Simen, meanwhile, said he has also reported the time resets to multiple groups at Microsoft. When reporting the problems on Microsoft's feedback hub in May, he said, he received no company response. He then reported it through the Microsoft Security Response Center in June. The submission was closed as a "non-MSRC case" with no elaboration.*

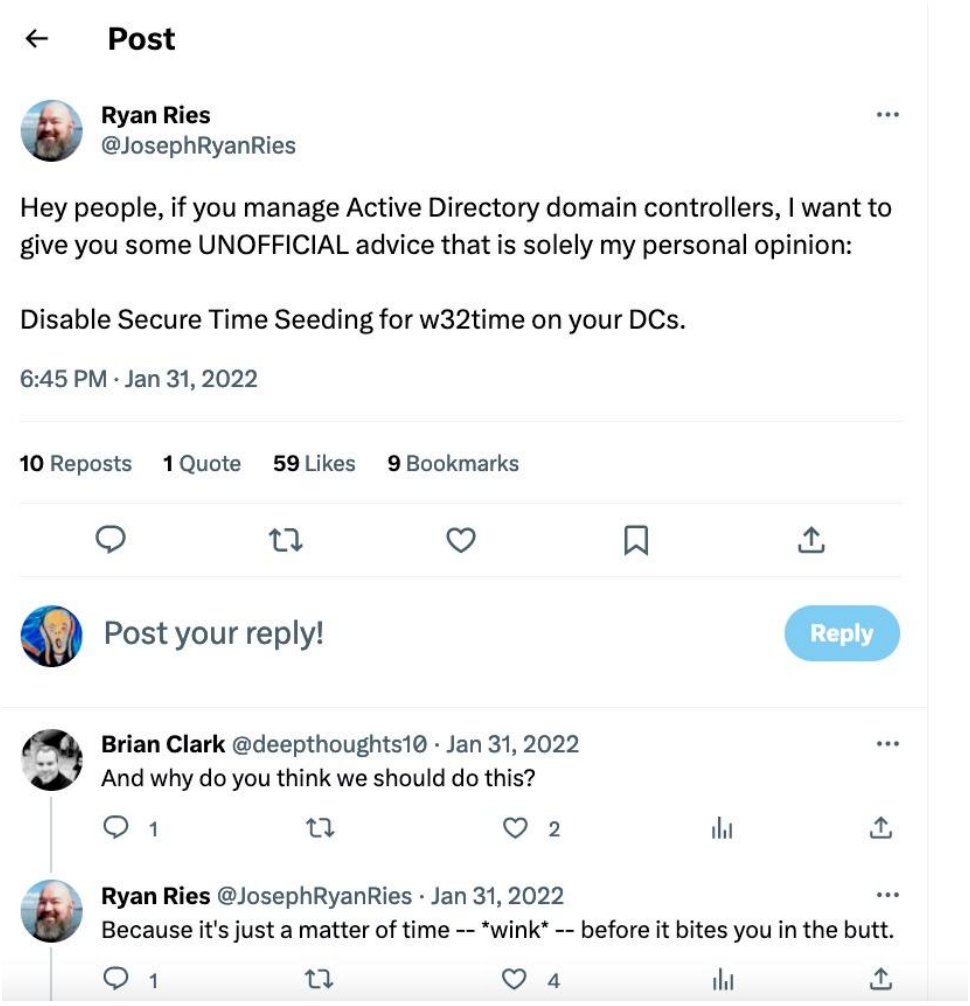
Simen then tapped a third party specializing in Microsoft cloud security to act as an intermediary. The intermediary relayed a response from Microsoft recommending STS be turned off when the server receives reliable timekeeping through the Network Time Protocol.

Simen wrote in an email: "Unfortunately, this recommendation isn't publicly available, and it is still far from enough to stop the wrongly designed feature to keep wreaking havoc around the world."

Simen said he believes the STS design is based on a fundamental misinterpretation of the TLS specification. Microsoft's description of STS acknowledges that some SSL implementations don't put the current system time of the server in the ServerUnixTime field at all. Instead, these implementations—most notably the widely used OpenSSL code library starting in 2014—populate the field with random values. Microsoft's description goes on to say, "We have observed that most servers provide a fairly accurate value in this field and the rest provide random values."

Simen said "The false assumption is that ***most*** *SSL implementations return the server time. This was probably true in a Microsoft-only ecosystem back when they implemented it, but at that time [when STS was introduced], OpenSSL was already sending random data instead."*

While official Microsoft talking points play down the unreliability of STS, Ryan Ries, whose LinkedIn profile indicates he is a senior Windows escalation engineer at Microsoft, wasn't as reticent when discussing STS on Twitter last year…



Again, this was a "senior Windows escalation engineer" at Microsoft. Several hours after Ars post appeared a Microsoft representative emailed Ars the following statement:

> *Secure Time Seeding feature is a heuristic-based method of time keeping that also helps correct system time in case of certain software/firmware/hardware timekeeping failures. The feature has been enabled by default in all default Windows configurations and has been shown to function as intended in default configurations.*
>
> *Time distribution is unique to each deployment and customers often configure their machines to their particular needs. Given the heuristic nature of Secure Time Seeding and the variety of possible deployments used by our customers, we have provided the ability to disable this feature if it does not suit their needs. Our understanding is that there are likely unique, proprietary, complex factors in deployments where customers are experiencing Secure Time Seeding issues and these customers do not benefit from this feature as it is currently implemented. In these isolated cases, the only course of action we can recommend is to disable this feature in their deployments.*
>
> *We agree that the overall direction of technology with the adoption of TLS v1.3 and other developments in this area could make Secure Time Seeding decreasingly effective over time, but we are not aware of any bugs arising from their use. This technology direction also makes heuristic calculation of time using SSL/TLS far less attractive when compared to deterministic, secure time synchronization.*
>
> *We continue to investigate how to best secure time synchronization on the Internet and welcome customer input on how to best meet their future needs.*

As Simen noted earlier, it's not clear precisely what causes STS to make the errors sometimes but not always. He wrote: *"This is what really strikes me as odd. Microsoft knows the field they look at might contain random data, so my guess is that their implementation breaks down when this is skewed so that most/all implementations they communicate with contain random data rather than just some."*

The well-known personality HD Moore, CTO and co-founder at runZero who is also a renowned network security expert, open source programmer, and hacker who founded the Metasploit Project and was the main developer of the Metasploit penetration testing software suite, speculated on Signal that the cause is some sort of logic bug in Microsoft code. HD wrote:

> *If OpenSSL has been setting random unix times in TLS responses for a long period of time, but this bug is showing up infrequently, then it's likely harder to trigger than just forcing a bunch of outbound TLS connections to a server with bogus timestamp replies—if it was that easy, it would happen far more frequently. Either the STS logic requires different root certificates as the signer, or some variety in the hostnames/IPs, or only triggers on certain flavors of random timestamp (like values dividable by 1024 or something). It smells like a logic bug that is triggered infrequently by fully random timestamps (32-bit) and likely just some subset of values and with some other conditions (like multiple requests in some period of time to multiple certs, etc.).*

Ars then wraps up their coverage of this by writing:

*As the creator and lead developer of the Metasploit exploit framework, a penetration tester, and a chief security officer, Moore has a deep background in security. He speculated that it might be possible for malicious actors to exploit STS to breach Windows systems that don't have STS turned **off**. One possible exploit would work with an attack technique known as Server Side Request Forgery.*

*Microsoft's repeated refusal to engage with customers experiencing these problems means that for the foreseeable future, Windows will by default continue to automatically reset system clocks based on values that remote third parties include in SSL handshakes. Further, it means that it will be incumbent on individual admins to manually turn **off STS** when it causes problems.*

That, in turn, is likely to keep fueling criticism that the feature as it has existed for the past seven years does more harm than good.

Simen wrote: "STS is more like malware than an actual feature. I'm amazed that the developers didn't see it, that QA didn't see it, and that they even wrote about it publicly without anyone raising a red flag. And that nobody at Microsoft has acted when being made aware of it."

We know that Simen doesn't listen to this podcast. He contacted me last week because others who do thought I'd be interested. If Simen did listen to this podcast he might wind up being somewhat less surprised by this behavior from Microsoft. Certainly, none of our regular listeners are surprised. Disappointed and saddened? Yes. Surprised? No.

A potential problem that HD Moore refers to is that there is now a very widely known bug in Windows timekeeping. And while, after seven years of neglect, Microsoft is clearly in no hurry to fix this, bad guys will likely be all over this code working out exactly how to trick Windows into mis-setting their clocks. And as Bruce Schneier says: *"Attacks never get worse, they only ever get better."*

In the registry, HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config Contains a REG_DWORD value named: UtilizeSslTimeData. That makes it pretty clear. And now that we know that the SSL time data that present in OpenSSL's handshakes is random noise, that doesn't seem like a good thing to leave enabled. In my Win10 machine that value was set to 1 to enable this feature of Windows. It's not set to '0' and anyone else who's concerned by this can set it to 0. After rebooting Windows will rely upon your local machine's clock and the reliable NTP Network Time Protocol.

As I started out observing: Heuristic "rules of thumb" approaches can be extremely valuable. They make SpinRite 6.1 possible. But they need to be designed with care and protected from going badly wrong. It appears that Microsoft has done neither of these things in a system that has already, and may still, damage their users.