Security Now! #957 - 01-16-24

The Protected Audience API

This week on Security Now!

What would an IoT device that had been taken over, do? And what would happen to the target of attacks it might participate in? What serious problem was recently discovered in a new post-quantum algorithm and what does this mean? What does a global map of web browser usage reveal? And after entertaining some thoughts and feedback from our listeners and describing the final touch I'm putting on SpinRite, we're going to rock everyone's world (and I'm not kidding) by explaining what Google has been up to for the past three years, why it is going to truly change everything we know about the way advertisements are served to web browser users, and what it all means for the future.

Please provide a clear visual example of "Irony"

Security News

What would an IoT device look like that *HAD* been taken over?

We've talked a lot about the threat that's posed by the remote takeover of IoT devices. We know, without any question, that there are a great many very large bot fleets, and that they are composed of individual unattended Internet-connected devices.

One of our listeners (Joe Lyon / @Joe_Lyon_) sent the image of a Twitter posting where the poster is rhetorically asking why his LG Washing Machine is using 3.6 gigabytes of data per day? And he attached an image that was produced by some network monitoring tool, showing that something on his network whose interface is labeled "LG Smart Laundry Open" is, indeed, quite busy with networking. And whatever is going on is happening very uniformly for a full 24 hours with only one hour out of the 24 showing a reduced total. So, yeah, that's certainly sufficient to raise suspicion. But what also caught my eye was that the labels on the traffic flow show a download of 95.75MB and a whopping upload of 3.57GB. Anyone who's worked with networking gear knows that it's important to know which directions "up" and "down" are referring to.

Cisco has always used the unambiguous terms "in" and "out" – as in, traffic flowing into or out of a network interface. If the interface is facing toward the Internet, then traffic flowing **out** of it would be toward the Internet and traffic flowing into it would be from the Internet. But if an interface is facing inward toward a local area network then the meaning of in and out would be reversed.

Without a bit more information about the network's configuration we cannot be 100% certain. But either the washing machine's networking system is badly broken, causing it to continuously download at a rate of three and a half gigabytes of ... something ... per day. Or, as does seem more likely given the evidence, and the label "Upload" even though we cannot be certain what that means, that this washing machine has become a bot in someone's army, so it's busy doing its part to flood targeted victims with nonsense traffic of some sort, to cause then grief at their end.

This brings me to two final observations: First, since the typical consumer is not monitoring their local network's traffic in any way, they would have no way to know that this was going on at all. And given the closed turnkey nature of an LG washing machine, it's unclear how one would go about reflashing its firmware to remove the bot. It might just be living in RAM. So pulling the plug, counting to ten, and powering it back up might be all that's needed. But then the device might become reinhabited again before long. So the only real solution would be to take the washing machine off the 'Net ... which nicely brings me to my second point:

What the heck is a washing machine doing being connected to the Internet in the first place?

This is another of those *"just because we can, doesn't mean we should"* situations. I've owned washing machines my entire life. The only thing any of them have ever been connected to is AC power. Is it really necessary for us to initiate a rinse cycle while we're out roaming around somewhere or to be notified with a message delivered through an app when our clothes are dry?

I get it that if all of that amazing functionality is free and included then why not set it up and get it on the Internet? But we're talking about this because of maybe why not. Maybe something has crawled into that machine and set up housekeeping. Maybe the only thing it's currently doing is flooding hapless remote victims with unwanted Internet traffic. But maybe also, if it wanted to, it could pivot and start poking around inside your residential network. And just maybe that could end up being a high price to pay for the luxury of being notified by an app when the lint filter needs to be changed.

If these sorts of things are going to be connected to your network, give some thought to sequestering them on a separate guest LAN which has no access to the high-value LAN. Most of today's consumer routers offer this feature. That makes it easier to implement than it was when we first started talking about the idea of LAN separation many years back.

And speaking of DDoS attacks

This related bit of news was also pointed to by a listener: Sukima @twit.social / @sukima, who wrote:

> *I use this service for all of my personal projects and liked it so much I was motivated to support them financially. And yet they are having a massive DDoS attack and thought it worth talking about publicly especially as examples of tech doing everything right and still being vulnerable.* https://outage.sr.ht/

So I went over and looked and wanted to share part of what I found. One of the three guys who run the services at SourceHut wrote:

> *My name is Drew, I'm the founder of SourceHut and one of three SourceHut staff members working on the outage, alongside my colleagues Simon and Conrad. As you have noticed, SourceHut is down. I offer my deepest apologies for this situation. We have made a name for ourselves for reliability, and this is the most severe and prolonged outage we have ever faced. We spend a lot of time planning to make sure this does not happen, and we failed. We have all hands on deck working the problem to restore service as soon as possible.*
>
> *In our emergency planning models, we have procedures in place for many kinds of eventualities. What has happened this week is essentially our worst-case scenario: "what if the primary datacenter just disappeared tomorrow?" We ask this question of ourselves seriously, and make serious plans for what we'd do if this were to pass, and we are executing those plans now – though we had hoped that we would never have to.*
>
> *I humbly ask for your patience and support as we deal with a very difficult situation, and, again, I offer my deepest apologies that this situation has come to pass.*
> ***What is happening?***

At 06:30 UTC on January 10th, two days prior to the time of writing, a distributed denial of service attack (DDoS) began targeting SourceHut. We still do not know many details – we don't know who they are or why they are targeting us, but we do know that they are targeting SourceHut specifically.

We deal with ordinary DDoS attacks in the normal course of operations, and we are generally able to mitigate them on our end. However, this is not an ordinary DDoS attack; the attacker possesses considerable resources and is operating at a scale beyond that which we have the means to mitigate ourselves. In response, before we could do much ourselves to understand or mitigate the problem, our upstream network provider null routed SourceHut entirely, rendering both the internet at large, and SourceHut staff, unable to reach our servers.

The primary datacenter, PHL, was affected by this problem. We rent colocation space from our PHL supplier, where we have our own servers installed. We purchase networking through our provider, who allocates us a block out of their AS, and who upstreams with Cogent, which is the upstream that ultimately black holed us. Unfortunately, our colocation provider went through two acquisitions in the past year, and we failed to notice that our account had been forgotten as they migrated between ticketing systems through one of these acquisitions. Thus unable to page them, we were initially forced to wait until their normal office hours began to contact them, 7 hours after the start of the incident.

When we did get them on the phone, our access to support ticketing was restored, they apologized profusely for the mistake, and we were able to work with them on restoring service and addressing the problems we were facing. This led to SourceHut's availability being partially restored on the evening of January 10th, until the DDoS escalated in the early hours of January 11th, after which point our provider was forced to null route us again.

We have seen some collateral damage as well. You may have noticed that Hacker News was down on January 10th; we believe that was ultimately due to Cogent's heavy handed approach to mitigating the DDoS targeting SourceHut (sorry, HN, glad you got it sorted). Last night, a non-profit free software forge, Codeberg, also became subject to a DDoS, which is still ongoing and may be caused by the same actors. This caused our status page to go offline – Codeberg has been kind enough to host it for us so that it's reachable during an outage – we're not sure if Codeberg was targeted because they hosted our status page or if this is part of a broader attack on free software forge platforms.

We were just talking about the LG Smart Washing Machine and the idea that it was apparently sending a continuous stream of traffic, totaling about three and a half gigabytes per day, out onto the Internet for some purpose. So I wanted to put a face on this to make it a bit more real for everyone. What I've just shared is a perfect example of where such traffic goes and its very real consequences.

Drew used the term "null routing" which is the action taken by major carriers, such as Cogent in this case, when some client, or client's client, or client's client's client is undergoing a sustained attack. They essentially pull the plug.

When an attack originates, as most do now, from a globally dispersed collection of bots, that traffic, which is all aimed at a single common IP address somewhere, will enter the network of a major carrier like Cogent across the globe as well. So that means that the attack is crossing into

Cognet's routers from all of its many various peering partners who are the ones whose networks have been infected with some bots, or perhaps the traffic is just transiting across their network and originates from some other major carrier's network. Whatever the case, the real danger of these attacks is their concentration. As the traffic hops from one router to the next, with each hop bringing it closer to its destination, traffic is being aggregated, is growing in strength, and can get to the point of debilitating the routers it is attempting to pass through.

This means that the optimal action for any major carrier, like Cogent, is to prevent this traffic aggregation by blocking the attacking traffic immediately at all and each of the many points of entry into their network from their peering partners. So Cogent sends routing table updates out to every one of the peering routers on their border, instructing that router to "null route" – meaning immediately discard – any packets attempting to enter their network which are bound for the IP that's under attack. This neuters the attack without causing any harm to their network. And since there will almost certainly be malicious bots inside the networks of some of Cognet's client ISPs, this null routing must also be applied internally as well as on their border.

But notice that now, with the targeted IP null routed, it's also impossible for any benign traffic to reach its destination service. A major carrier's null routing not only blocks the attacking traffic, but any and all traffic to that service – good or bad. In fact, once the attack has subsided and full service could be resumed, the site will remain dark until someone at the service provider notices that the attack has mitigated and lifts the network-wide block to allow regular services to resume.

DDoS attacks like this one have become a fact of life on the Internet. Anyone who is working for any major service provider sees them and deals with them now as part of their daily routine. But as we've just seen, this doesn't make such attacks any less significant and potentially devastating to anyone who is committed to keeping whatever services they offer available.

And we also know where these attacks originate. They originate from devices exactly like that LG Smart Washing Machine... a gadget that largely operates autonomously where networking is not its primary focus. It was tacked on as a feature so it never got the attention that it needed to be a truly secure networking device. And we also know that the phrase "truly secure networking device" almost needs to be said with tongue in cheek because, sadly, it has become an oxymoron. Say it slowly... "Truly Secure Networking Device". It's almost become the Holy Grail. Everything we've learned is that it is truly difficult to create and maintain a truly secure networking device. And the more features are added, the more quickly the challenge grows.

Trouble in the Quantum Crypto world

BleepingComputer recently reported the news that many implementations of the already-in-wide-use post-quantum key encapsulation mechanism known as "Kyber", which is in use, for example, by the Mullvad VPN and to provide Signal's post-quantum redundancy, has been found to be subject to timing attacks. The set of attacks have been dubbed "KyberSlash".

The first thing to understand is nothing has been found wanting from the post-quantum Kyber algorithm itself. As far as anyone knows, Kyber still provides very good quantum resistance. The problem and vulnerability is limited to some of the actual code that was used to implement

the Kyber algorithm. And this is part of the typical shaking out process that new algorithms undergo. First we need to get the theory right. Then it's tested to prove that it does what we thought. Next, the code is implemented into libraries where it can actually be used and tested in the real world. And it was at this point in the process that these troubles arose.

The problem is that the vulnerable algorithms perform an integer division instruction where the numbers being divided are dependent upon the algorithm's secret key. Since division is an instruction whose timing can vary widely with the binary bit pattern of the specific numbers being divided, this naturally results in a situation where the secret that needs to be kept has the potential to leak out through a timing side-channel. And that's exactly what happened here.

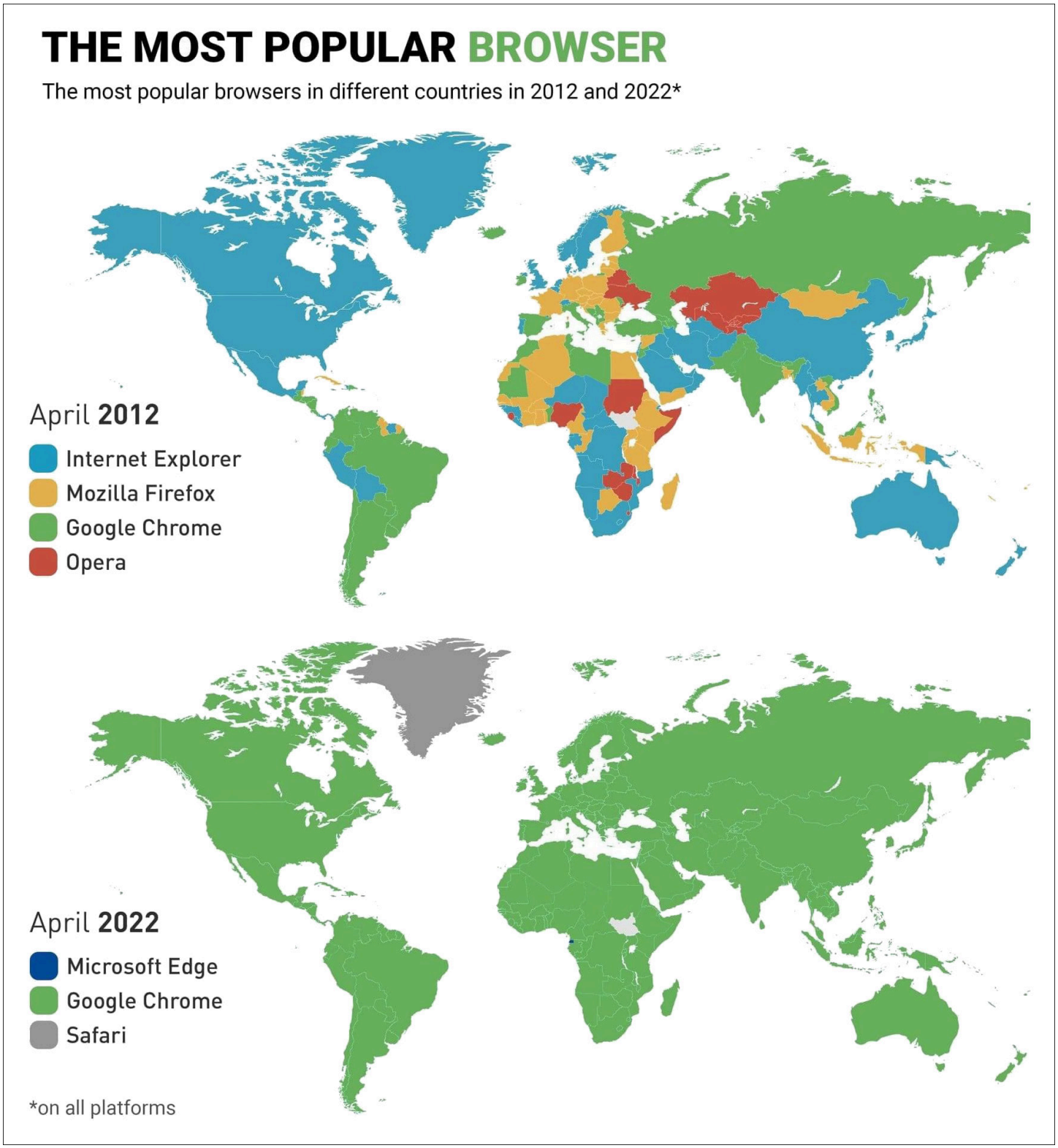
It's such an obvious and well understood problem that whoever wrote that code should be scolded. Perhaps they were just having a bad day. Or perhaps they're solely focused on theory and not enough on practice. Who knows. But in any event, the problem was found, it's well understood, and many of the libraries that implemented the vulnerable reference code have been updated and more are being updated.

It's a good thing that we're doing this now rather than two years from now when we might have actually been dependent upon the strength of these algorithms to protect against quantum based attacks.

The Browser Monoculture

StatCounter produced this bracing snapshot of global web browser use 12 years ago, back in 2012 and two years ago in 2022. Since today's podcast is all about Google and their Chrome browser, this seemed quite apropos.

The map speaks for itself but it's a bit breathtaking:



Closing the Loop

There were, predictably, many replies from our listeners about my follow-up discussion last week of the Apple Backdoor. I've just grabbed one to finish out this subject...

David Wise / @davelwise

Hey Steve. Listening to your podcast about the Apple vulnerability. Could this be a supply chain hack with the phones being built in China?

Sure. Absolutely. Unfortunately... literally anything is possible. I think it's safe to say that by the nature of this we'll never have a satisfactory answer to the many questions surrounding exactly how or why this all happened. All we know for sure is that backdoor hardware exists in the hardware that Apple sold and shipped. And notice by David's question that plausible deniability exists. All of the several possible sources of this can claim absolute surprise and total ignorance of how this came to pass. Is it significant that it first appeared in 2018, three years after the famous high visibility 2015 San Bernardino terrorist attack? Several years being the approximate lead time for a new silicon design to move all the way through verification, fabrication, and into physical devices? Again, we'll never know... and, yeah, that's annoying.

osyncsort / @osyncsort

Hi Steve, I'm a fan of the podcast and all the great work you have done in your career (especially a fan of sqrl. Thank you for this great work and I wish it to be mainstream in the near future). I recently listened to episode 885 and you briefly touch on a subject that I have been contemplating(getting into infosec). I'm currently thinking about getting into infosec as a career. I'm in my 40s and wanted to know, from your perspective, if 40s is too old to get into the field? My career is online marketing and I've been fortunate to have been doing it from the early days of Web 1.0 to what is now referred to Web 3.0. However After Covid(2020) I have not had the same opportunities in the marketing world. So I find myself looking for a new career and thinking infosec may be the solution. Any advice/option is welcomed. Thanks in advance.

The good news is that there is a huge and unmet demand for information security professionals. I think that the trouble with any sort of "I am too old" question is that so much of the answer to that depends upon the individual. A particular person in their early 30's might already be too, whereas someone else who's 55 might not be. But speaking only very generally, since that's all I can do here, I think I'd say that someone "In their 40's" probably spans the range where "too old" might start to be an issue if it's a concern for them. Early 40's?, not so much. Late 40's?, maybe a bit moreso. But, regardless, there's definitely something (a lot actually) to be said for having some gravitas and life experience that only comes with age. An IT guy who is more world-wise will generally be much more useful than a fresh newbie who is still addressing the world with impractical expectations. And especially for an IT guy, knowing how to talk to others is a skill that should not be undervalued. So I think that on balance I'd say "go for it" and know that the demand for that skill set will only be increasing over time.

3n0m41y / @3n0m41y

Hi Steve, I would like to get your opinion on Proton Drive Vs Sync as a secure cloud storage network. Recently the Sync iOS app has broken the ability to natively use the built-in iOS FILES app to navigate sync's folder structure properly. What happens is after drilling down 1-2 directories, the sync app pushes the structure back to the root folder. This while not a show stopper, it does break the use of other 3rd party apps on iOS. I have reached out to the Sync dev team but they've responded that it will take "Quite a while" to fix. This functionality broke about 2 months ago. So I just want to get your take if Proton has matured enough to be a replacement for Sync… Cheers and Happy New Year!

So, first of all, let me just note that I'm very disappointed when something I deeply research and then strongly endorse (anyone remember LastPass) later evolves – or devolves – in such a way that I come to regret my previous full throated endorsement. So I'm disappointed to learn that Sync is not standing behind and repairing their various client apps.

As for Proton Drive, I haven't looked into it deeply enough to form any opinion one way or the other. However, its "Proton" name should be familiar, since these are the same Swiss people who offer the very popular Proton Mail service. So my strong inclination would be to trust them. What I have no idea about is how feature rich their offering might be. But my shooting from the hip thought would be that if it does what you want, for a price that makes sense, I'd say that based upon their past performance on everything else we know they have done, I'd be inclined to give them the benefit of any doubt. That's obviously not definitive, but at least it's something.

SpinRite

It has been so very useful these past final months of work on SpinRite to have this podcast's listeners taking the current SpinRite release candidate out for a spin, and providing feedback about their experiences. That feedback has been the primary driving force behind the last few improvements to SpinRite, which turned out to be significant. So I'm glad that I didn't declare it finished before that. And it's been a slowly growing chorus of feedback that caused me to decide that I needed to change something else.

If you've been following along, you'll recall that one of the astonishing things we discovered during SpinRite's development was that all of the original and past versions of the PC industry's most popular BIOS, produced by American Megatrends, Inc. and commonly known as the AMI BIOS, contain a very serious bug in its USB handling code: Any access to any USB-connected drive past the drive's 137 gigabyte point – which is where 28 bits of binary sector addressing overflow into the 29th bit – causes these AMI BIOSes, which are in the majority, to overwrite the system's main memory right where applications typically load.

I was so appalled by that discovery, and by the idea that this could be very damaging, not only to SpinRite's code but potentially to its users data, that I decided to flatly prohibit SpinRite v6.1 from accessing any USB-connected drive past its 137 gigabyte point. The next SpinRite won't suffer from any of this trouble since it will have its own direct high-performance native USB drivers. So my plan was to just get to SpinRite 7 as quickly as possible. But SpinRite's early users who have attached a larger-than-137 gigabyte drive via USB, and then had 6.1 tell them that it could only safely test the first 137 gigabytes of their drive, have not been happy.

Also, since then, one of the guys who hangs out in GRC's newsgroups, Paul Farrer, was curious to learn more about this. So he looked into the problem while I continued to work on other aspects of SpinRite. Paul wrote a set of exploratory DOS utilities and tested them with a bunch of old motherboards owned by some of our SpinRite testers. What he discovered suggested that more could be done than just turning my back on all USB BIOS access in disgust. And the disappointment I was seeing from new people being exposed to SpinRite's refusal to work with any USB BIOS convinced me that I needed to fix this. So I started on that last week and I expect to have it finished this week.

Since only the AMI BIOS is known to have this problem, SpinRite will start by lifting this blanket ban from all non-AMI BIOSes. Then, for AMI BIOSes, since they don't all have this trouble, I've reverse-engineered the code surrounding the bug and now I fully understand what's going on. So SpinRite can now detect when the bug is actually present and can patch the buggy BIOS code in place, to raise SpinRite's access limit from 137 gigabytes to 2.2 terabytes. The buggy AMI USB BIOS code will still have a bug that prevents it from working past 2.2 terabytes, but that's much better in today's world than clamping all USB BIOS access, for everyone, at 137GB.

The Protected Audience API

I mentioned last week that I thought I might be onto an interesting topic to explore this week. It turned out that while the guy I stumbled upon was the real deal, his several blog postings were sharing pieces from his Master of Law dissertation for the University of Edinburgh. After I looked into it more deeply, it didn't really make for the sort of content I know our listeners enjoy. This scholar was carefully examining the **legal** and **policy** implications of Google's recent work on the web – the set of new technologies collectively known as "The Privacy Sandbox" – against EU & UK laws like the GDPR. And this guy was not just some lawyer. He's a deep technology guy who has been actively involved with the W3C, serving on many committees and having co-authored a number of web specs. His focus has always been on privacy and he's the guy who, years ago, realized that the addition of a high resolution "battery level meter" into the HTML5 specifications would provide another signal that could be used for fingerprinting and tracking. But as I said, his focus was on what Google's recent work would mean legally. And for what it's worth, this very well informed legal and technical academic, who is also a privacy nut, is quite bullish on the future Google has been paving for us. So that just means that what we **are** going to talk about this week is all the more relevant and significant.

And what we **are** going to talk about this week is something known as the "Protected Audience API". It's another of the several components which make up what Google collectively refers to as their "Privacy Sandbox." The name Protected Audience API is every bit as awkward as many of Google's other names. They're a big company. They could afford to employ someone with the title of *"Director of Naming Things"* – and give this person a big office and a staff, because it's clear (and it will soon become much clearer) that the nerds who invent this stuff should not be the ones to name it. In this instance, what's "protected" is user privacy and "audience" refers to the audience for web-based display advertising. But as it is, calling this the Protected Audience API only tells you what it is after you already know; which is not the definition of a great name.

In any event, this collection of work that Google has called their Privacy Sandbox currently contains a handful (dare I say "plethora?") of different APIs. There's the new **Topics API** which we've previously covered at length. And there's the **Protected Audience API** which is what we'll be looking at today. But then there's also something known as the **Private State Tokens API**, the **Attribution Reporting API**, the **Related Website Sets API**, the **Shared Storage API**, the **CHIPS API**, the **Fenced Frames API** and the **Federated Credential Management API**. And if you didn't already know what those things are, knowing their names only helps with the very broad strokes. But here is what everyone listening really does need to know: All of this is brand new, serious, deliberately user privacy focused technology which Google's engineers have recently created and unfortunately named. But collectively it represents a truly **major** step forward in web technology.

We all grew up in, and cut our teeth on, extremely simple web technology that its founders would still clearly recognize today. Even after many years, this baby hadn't grown much and it was still far from mature. We had cookies and JavaScript and ambition, and a lot of ideas about what we wanted to do with the web. But everyone was heading in their own direction, doing whatever they needed for themselves just to get the job done, and no one was thinking or worrying about longer term consequences.

The WEB lacked the architectural and technological depth to get us where we wanted to go in the way we needed to get there. So we wound up with the absolute chaos of tracking and identity brokering and personal data warehousing, de-anonymizing and all the rest of the mess that defines today's world wide web. And example of the mess we're in today is the utter pointless bureaucratic insanity of the GDPR forcing all websites to get cookie usage permission from each of their visitors.

We know that Google is fueled by the revenue generated from advertising. Advertisers want to know everything they possibly can about their audience. They want to optimize their ad buys. And users are creeped-out by the knowledge that they're being tracked around the Internet and profiled. And being the super heavyweight that it is, Google is increasingly under the microscope. But they also have the technological savvy – probably unlike most other players on Earth at this time in our history – to actually solve this very thorny problem which arises from the collision of apparently diametrically opposed interests on today's web. One thing is clear: We're in desperate need of more technology than cookies.

Google began the work to truly solve these problems, in earnest, three years ago at the start of 2021. And this wasn't some half-baked attempt to gloss over the true problems that are inherent in the use of a system that was never designed or intended to be used as it is being used today.

Google's Privacy Sandbox initiative was, and today is, a significant major step forward in web browser technology and standards which is designed to allow the web to finance its own ongoing existence and services through advertising, ***without in any significant way compromising the privacy of its users.***

We have all been so badly abused by the way things have been that it may be difficult to accept that there truly is a way for this to be accomplished. But there is; and Google has done it. In the future, the use of the web will be much more private than it has ever been since it was first conceived. What's required to make that possible is way more technology than has ever been deployed before. What's been done before now couldn't even be called a half measure.

All of the various APIs mentioned above (whatever it is they each do) became available in the middle of last year at the start of the 3rd quarter of 2023. They are all operable today, right now, in the world's dominant web browser and the other browsers that share its Chromium engine.

And it's not as if there weren't some wrong turns made along the way. But that's also the nature of pioneering where the path hasn't already been mapped out. FLoC – Google's Federated Learning of Cohorts was an attempt at generating an opaque token that revealed nothing about the browser's user other than a collection of their interests. But FLoC didn't get off the ground. It was later replaced by TOPICS which is a complex but extremely clever system for doing essentially the same thing – but in a far less opaque and thus far more understandable fashion.

TOPICS allows the user's browser to learn about the user by observing where they go on the web – all of which information is retained by and never leaves the browser. Then, through the use of the ***Protected Audience API***, the user's **browser** is able to later intelligently select the ads that its user will see. If that comes as something of a surprise, it should, since it's certainly not the way any of this has ever worked before.

One of the key features to note and to keep in mind is that this expands the role of the web browser significantly. There is now far more going on under the covers than ever before. It was once fun and easy to explain how a web browser cookie worked. It wasn't difficult to explain because there wasn't much to it. But there is very little that's easy to explain about how these various next-generation Privacy Sandbox browser APIs function. And this is made even more difficult by the fact that they're all so deeply interconnected. When we originally discussed TOPICS we had no sense that its purpose was to allow the user's browser to autonomously perform ad selection – but that was always Google's intent; we just needed to see more of the whole picture. And even when we were only seeing a small portion of the whole, explaining the operation of TOPICS required a very careful description because it is laced with important subtleties. And I suppose that's the main point I want to convey here. Because we're now asking so much from the operation of the web – even wanting things that appear to be in direct opposition – the simple solutions of yesterday will not get us there.

So what is this Protected Audience API? Believe it or not, opaque as even that name is, the good news is that they renamed this to the "Protected Audience API". Which, of course, begs the question "Renamed it from what?" Recall that the earlier, abandoned, interest-gathering API was originally given that awkward name "FLoC" which stood for Federated Learning of Cohorts. (Ouch.) In a similar vein, the Protected Audience API was originally named FLEDGE, a painfully reverse-engineered acronym which stood for: **F**irst **L**ocally-**E**xecuted **D**ecision over **G**roups **E**xperiment. Not an exactly catchy name. Where's the Director of Naming Things when you need them?

And what you're really not going to believe is that FLEDGE grew out of a project named "TURTLEDOVE". I kid you not. And, yes, TURTLEDOVE was also an acronym, short for: **T**wo **U**ncorrelated **R**equests, **T**hen **L**ocally-**E**xecuted **D**ecision **O**n **V**ictory – they're only missing a word to provide the "E" at the end of DOVE. "Excellent?" "Everlasting?" How'bout "Excruciating?"

Anyway... I was able to explain how TOPICS worked since while it was a bit tricky and subtle, it was a relatively self-contained problem and solution. I don't have that feeling about this Protected Audience API because, as I noted earlier, they each only really make coherent sense when they're taken as a whole. So I'm not going to attempt to explain it at the same level of transactional detail. But I want to at least share some sound bites so that you can come away with some sense for what's going on here.

At the start of Google's Protected Audience API explainer page, it opens with one sentence that needs to be taken absolutely literally. Listen to this:

> **On-device ad auctions to serve remarketing and custom audiences, without cross-site third-party tracking.**

"On-device ad auctions". Okay. Now, I don't expect anyone to understand in any detail what follows. I certainly don't. So just let it wash over you and you'll get some very useful feeling for what's going on. Google "explains"...

> ***The Protected Audience API*** *uses interest groups to enable sites to display ads that are relevant to their users. For example, when a user visits a site that wants to advertise its products, an interest group owner can ask the user's browser to add membership for the interest group. If the request is successful, the browser records:*
>
> - *The name of the interest group: for example, 'custom-bikes'.*
> - *The owner of the interest group: for example, 'https://bikes-r-us.example'.*
> - *Interest group configuration information to allow the browser to access bidding code, ad code, and real-time data, if the group's owner is invited to bid in an ad auction.*

I know... just let your head spin. It'll be okay. So there is a feeling of the way TOPICS works here. The key is that the user's browser visits a site like "custom bikes" and because their browser is at that site, thus the user is implicitly expressing their interest in "custom bike", an advertiser **on that site** can ask the user's browser to collect and retain some information that might be used in the future if an ad from that advertiser will be displayed. Note, importantly, that the advertiser learns exactly nothing about the visitor to the site. All of the information flow is into the user's browser, and only because of the website they're visiting. Google & I continue...

> *Later, when the user visits a site with available ad space, the ad space seller, either a sell-side provider (SSP) or the site itself, can use the* ***Protected Audience API*** *to run [a browser-side] ad auction which will select the most appropriate ads to display to the user. The ad space seller calls [the browser's new]* ***navigator.runAdAuction()*** *function, to provide the browser with a list of interest group owners who are invited to bid.*
>
> *Bids can only be provided by interest groups that the browser already became a member of when it had previously visited a website where it was able to collect that group, and when the owners of those interest groups have been invited to bid. Bidding code is retrieved from a URL provided in the interest group's configuration that was received earlier. This code provides data about the interest group and information from the ad seller, along with contextual data about the page and from the browser.*
>
> *Each interest group providing a bid is known as a buyer.*
>
> *When the visited website's JavaScript calls the new browser function to run the ad auction, each buyer's bidding code generates a bid with the help of real-time data provided by their Protected Audience Key/Value service [whatever that is]. Then, the advertising space seller receives these bids as well as seller-owned real-time data and scores each bid. The bid with the highest score wins the auction.*
>
> *The winning ad is displayed in a fenced frame [which absolutely prevents it from having any interaction with anything else anywhere]. The ad creative's URL is specified in the bid, and the origin must match one in the list provided by the interest group's configuration – that same information that was received earlier.*

Finally, the advertising space seller can report the auction outcome (with the reportResult() function), and buyers can report their auction wins (with the reportWin() function).

A bit later, Google offers a bit more detail, writing:

In the Protected Audience API, an ad auction is a collection of small JavaScript programs the browser runs on the user's device to choose an ad. To preserve privacy, all ad auction code from the seller and buyers is run in isolated JavaScript worklets that cannot talk to the outside world.

A seller (a publisher or a supply-side platform) initiates a Protected Audience ad auction on a site that sells ad space (such as a news site). The seller chooses buyers to participate in the auction, indicates what space is for sale, and provides additional criteria for the ad. Each buyer is the owner of an interest group.

The seller provides the browser with code to score bids, which includes each bid's value, the ad creative URL, and other data returned from each buyer. During the auction, bidding code from buyers and bid-scoring code from the seller can receive data from their Key/Value services. Once an ad is chosen and displayed (in a fenced frame to preserve privacy) the seller and the winning buyer can report the auction result.

If all of this sounds insanely complex, you're right. This is not your grandpa's 3rd-party cookies anymore. Nor are our web browsers simple apps running on our chosen OS to display HTML code. Those are the days that are long gone and they're not coming back. It should now be abundantly clear to everyone that what Google has done with this Privacy Sandbox is to radically transform our web browsers from passive displays of whatever page is sent to them, into proactive advertising management engines. All of this new technology is built into Chrome and has been for the past 6 months.

Does all of this probably give Sir Timothy John Berners-Lee – the web's original inventor – a huge headache? I would not be at all surprised if it did. Nothing less than an incredible mess is required to deliver interest-driven advertising to users without revealing anything about those users to their advertisers. And by the way, "An Incredible Mess" was the runner-up title for today's podcast. A large part of what I want to convey here is that nothing short of this level of complexity is required to protect our privacy while providing what the websites we depend upon and want unpaid access to, say they need.

Now, the nature of inertia means that we would never – and I really mean never – move from the absolute mess we're in today to this new promised land were it not for a behemoth like Google to, first, carefully design and craft this solution — doing so openly and in full public view, inviting collaboration and industry participation at every step of the way, as they have. And secondly, to then literally **force it down the closed, choking throats** of the rest of the existing advertising technology industry by taking Chrome, their world domineering browser, and gradually deprecating and foreclosing upon the operation of **all** of the previous tricks and techniques that have historically been used for user tracking users and compromising their privacy in the service of advertising technology.

No one else could do this but Google. This is not something where consensus could ever be reached. It would never happen. It would be "committee deadlock". I've looked at the various ad tech blogs and they're all screaming and pulling their hair out over this... but they're all also busily conducting experiments and getting ready for what they, too, understand is already inevitable.

Notice that one of the things Google has done with this reconceptualization of web advertising is to move the advertising auctioning process away from the advertiser and into the browser. Traditionally, an advertiser would purchase real estate on the web on website pages. Then they would run their own real time auctions to determine which of their many advertising clients' ads should be inserted into that space for any given visitor given everything that the advertising seller knows about the visitor from tracking them around the Internet. This changes all of that.

Now, all of that work is being done on the client side rather than on the server side, and doing this starves advertisers of all the data they were previously collecting while convincingly arguing against their having any further need to ever collect anything again. In this new world, advertisers place static purchase offers to display content on website real estate with whatever ads they have to display, organized by interest group. Using Google's new APIs, browsers that had previously visited websites representing various interest groups are now able to collect the advertiser's material that will later be needed to display ads for those interested. Then later, when browsers visit websites with sell offers behind available advertising real estate, all of the information about the offers flows to the browser, which then itself, conducts the auction and selects the ad that is most relevant to its user based upon the places the browser has visited during the past few weeks.

The results of the auction are returned to all interested parties and the ad tech company pays a piece of the action (or the auction) to the site that offered up the real estate. In something of a follow-up, Google explains:

> *Understanding user interests can enable more relevant ads than just choosing ads based on site content (contextual targeting) or by using information provided by a user to the site on which the ad appears (first-party data targeting).*
>
> *Traditionally, ad platforms have learned about user interests by tracking their behavior across sites. Browsers need a way to enable ad platforms to select relevant ads, so content publishers can get ad revenue without cross-site tracking.*
>
> *The Protected Audience API aims to move the web platform closer to a state where the user's browser on their device—not the advertiser or ad tech platforms—holds information about what that person is interested in.*

And that states it perfectly, I think:

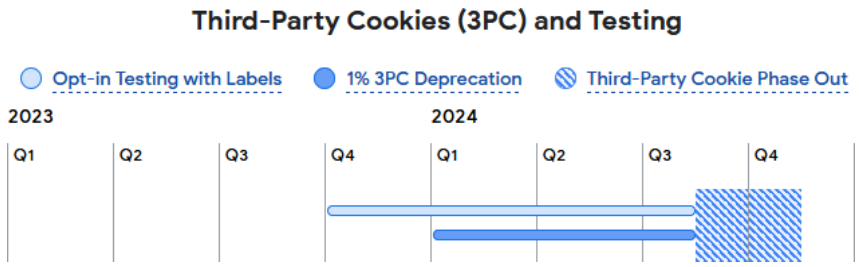
The way the entire web advertising world has worked until now is that every advertiser had to collect all of the information they possibly could about every individual who was surfing the Internet for the sole purpose of selecting the best advertisement to show them. The result was massively intrusive, massively redundant, and an ultimately inefficient utilization of resources.

But in the new world of Google's Privacy Sandbox, it's the user's browser that collects the information about its own user's interests by watching them navigate the web. As the browser moves around the web, future advertising opportunities are collected by the browser. And later, when visiting a site that is offering some available advertising space, the browser runs an auction to decide which of the opportunities it previously collected should be presented to its user based upon criteria that **it**, solely, maintains.

This is obviously a big deal. But what seems just as obvious is that no lesser of a deal would get this important job done right. We can argue, and we'll always be able to argue – we certainly know that the EFF will always argue – that all website user-driven advertising customization should simply be ended, and that advertisers should settle for contextual advertising – placing their ads on sites which are offering content that's relevant and related to their ads – just like in the pre-tracking days. Unfortunately, multiple studies have shown that this would reduce website advertising revenue by half, and many websites are barely making ends meet as it is. So the EFF's ivory tower stance is simply not practical – and it's never going to happen.

The only way to permanently end tracking is for it to be flatly outlawed. But tracking will never be outlawed while the case can be made that advertising customization is the only thing that's keeping today's web alive and financed, and that there is no alternative to tracking and compiling interest-profiling dossiers on everyone using the Internet. So what Google has done is to create a practical and functioning alternative. Tracking is no longer necessary, user privacy is preserved, and once this new system has been established we can anticipate that we will finally see legislation from major governments – probably with Europe taking the lead – which will flatly and without exception outlaw any and all Internet user profiling and history aggregation.

Google's Privacy Sandbox masterpiece has been in place for the past six months and although they have been kicking and screaming, all other serious advertisers have been exploring it in anticipation of the future which appears to be all but certain. As we move into 2024, fingerprinting will become increasingly fuzzy and Chrome's 3rd-party cookie support will be gradually withdrawn from its ubiquitous web browser.



And once the dust settles on all this, we can anticipate the end of the annoying cookie permission request pop ups!