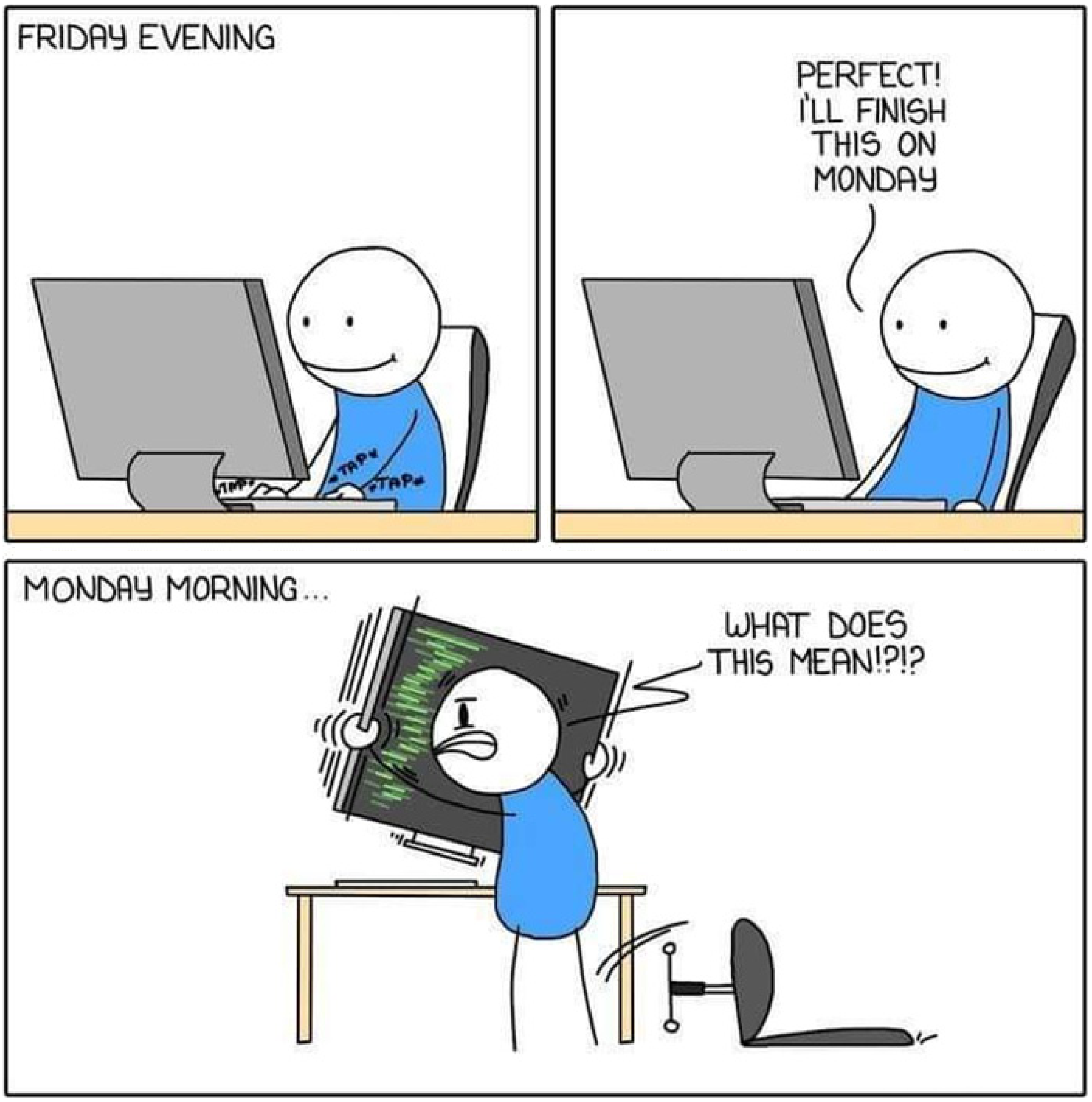
Security Now! #873 - 05-31-22
DuckDuckGone?

This week on Security Now!
This week we examine the difficult-to-believe-in-2022 design of Australia's New South Wales Digital Driver's License which was sold as being quite difficult to counterfeit. We examine the latest, once again fumbled, extremely pervasive Microsoft Office 0-day remote code execution vulnerability. We look at the first instance of touchscreen remote touch manipulation, and at Vodafone and Deutsche Telekom's difficult to believe yet already being piloted plan to further monetize their customers by somehow injecting persistent super-cookies into their customer's connections at the carrier level. Then, after sharing some feedback from our terrific listeners, we'll dig into the discovery that the DuckDuckGo Privacy Browser carved out a privacy exception for Microsoft.

Been there...



Security News

New South Wales DDL — Digital Driver's License — (WCPGW?)
This piece was stiff competition for this week's main story. But it lost out to an explanation of what was discovered about the operation of DuckDuckGo's Privacy Browser. So I decided to lead off with this one as the runner up.

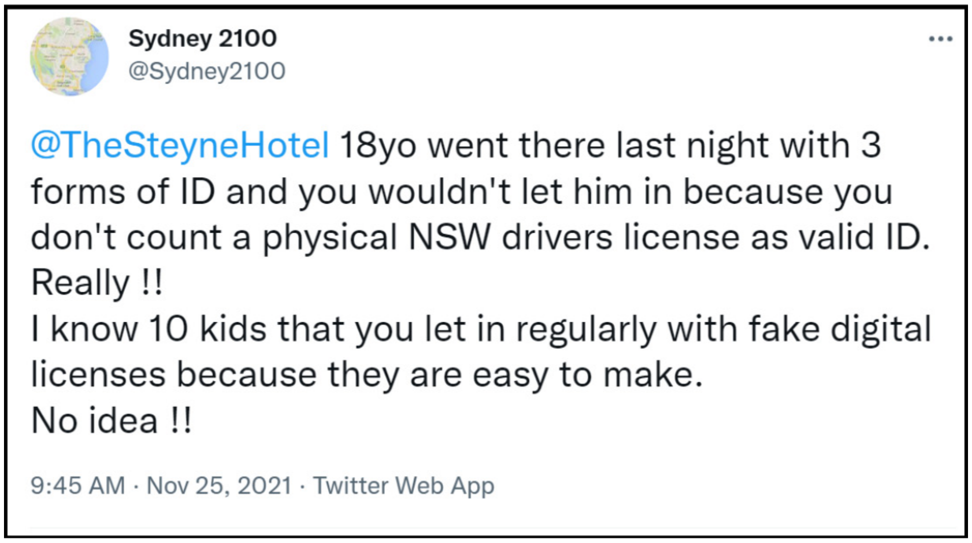
A penetration testing and secure app development group known as “Dvuln” recently took a close look at the New South Wales government's Digital Driver's License (DDL) and putting it mildly, found its security to be wanting. They documented the system's various troubles in a blog posting titled: “ServiceNSW’s Digital Drivers Licence Security appears to be Super Bad.”

To set the stage, they explain:

> *In November 2019, the New South Wales government (ServiceNSW) introduced the digital driver's license or “DDL” for short, as a means to make it easy for people to access a digital version of their driver license.*
>
> *Upon the launch of ServiceNSW’s Digital Driver License there were multiple security researchers who publicly reported a number of security issues including but not limited to the ability to manipulate Digital License data and create fraudulent digital identities.*
>
> *As far as we can see, there appears to be no formal public response from ServiceNSW regarding the acknowledgement or remediation of such issues.*
>
> *As of February 2022, according to the Minister for Customer Service there have been 3.9 million people who have opted-in for the Digital Driver Licence.*
>
> *To put this into perspective, we can assume around 70% of people in NSW use and trust the digital driver's license as a means of identification and verification in their day to day lives.*
>
> *During Dvuln’s analysis of the ServiceNSW mobile application (iOS), we discovered that due to the existence of several secure design flaws, it is still possible for malicious users to generate fraudulent Digital Driver's License with minimal effort on both Jailbroken and non-jailbroken devices without the need to modify or repackage the mobile application itself.*

Back in 2019, not long after the DDL first appeared, during a security conference a security researcher demonstrated his ability to modify DDL details locally on his mobile device, causing it to show false information. And although during his talk he mentioned that he had reported these troubles to ServiceNSW, there were no apparent public updates on the matter. So it's unclear whether these bugs were considered an accepted risk, or if any sort of remediation was ever attempted by ServiceNSW.

So now we jump forward three years to 2022 where there are rumors circulating regarding underage people using fake digital licenses. The Dvuln posting contained an authentic appearing twitter posting made on November 25th, 2021 where the poster is annoyed that a bouncer at a club denied access to one 18 year old when others using fake digital licenses are regularly admitted:


Sydney 2100
@Sydney2100

@TheSteyneHotel 18yo went there last night with 3 forms of ID and you wouldn't let him in because you don't count a physical NSW drivers license as valid ID.
Really !!
I know 10 kids that you let in regularly with fake digital licenses because they are easy to make.
No idea !!

9:45 AM · Nov 25, 2021 · Twitter Web App

Not having seen the Ids themselves, the Dvuln guys wrote:

We cannot confirm whether or not they were exploiting the poor security design or simply using a static photoshopped image, although due to the ease of exploitation, it is entirely possible that these kids were using the same method detailed in this blog.

The Dvlun guys quoted one of the security claims made about the new Digital Driver's Licenses:

1. According to a press release from the New South Wales government, the Digital Driver Licence implementation is "hosted securely on the new Service NSW app, locks with a PIN and can be accessed offline", and "will provide additional levels of security and protection against identity fraud, compared to the plastic driver license". -

And they explained that, in fact, real world physical driver's license counterfeiting is actually far more difficult that spoofing the content of a physical driver's license. They wrote:

Given the Digital Driver License's current state of security [meaning lack of security], *we believe it would be far more difficult for an average fraudster to obtain the equipment necessary to produce high quality plastic NSW drivers licenses. A fraudster would need to source and obtain hardware such as but not limited to:*
- *A card printer*
- *NSW holographic security foil*
- *Other security features developed uniquely for the NSW identification cards such as the middle green layer none of which are commercially or legally available outside the printing hardware.*

So what are the specific problems with New South Wales DDL's?

First off, the DDL-stored license data lacks secure encryption:

On iOS, the Digital Driver Licence data is stored in a JSON formatted file, which is encrypted using AES-256-CBC (Cipher Block Chaining mode) combined with Base64 encoding. And there's nothing wrong with that. Encryption will turn anything, even if it was ASCII textual content to begin with, into binary data. So the Base64 encoding converts that back into ASCII text so that it can be stored in a JSON text format file.

But here's the problem: The encryption key is a 4-digit PIN that's initially set during onboarding when the user first installs the application. That's it. A 4-digit PIN. Believe it or not, that's the password. A 4-digit PIN. Since a 4-digit numeric PIN can be anything from 0000 to 9999 there, gee, let’s see... ah, 10,000 possible PIN combinations... one of those will correctly decrypt the JSON file. And the use of the correct decryption will be readily apparent because what’s decrypted will not be gibberish. In other words, if an attacker is able to obtain the encrypted data, either through accessing an iPhone backup, direct access to the device or a remote compromise, it will only take a few minutes to brute-force the correct 4-digit key. There is not even any password-based key derivation function (PBKDF) to slow down the guessing.

During Dvuln's testing, their brute-forcing process only took a few minutes to decrypt the Digital Licence Data which could then be edited, re-encrypted and used to change the Digital Driver Licence details on the mobile device.

Unbelievable. No sign of any authentication, no digital signature or any other protection against user manipulation of the stored and then displayed data.

The next problem is a lack of client-side validation... The Digital Driver Licence data is never validated against the back-end authority which issued the license. This means that the application has no native method to validate the Digital Driver Licence data that exists on the phone and thus cannot perform further actions such as warn users, or others, when this data has been modified. Since the digital driver's license data is stored on the client's device, validation should take place to ensure the local copy of the data matches the Digital Driver's Licence data that was originally downloaded from the Service NSW API. But since no such verification takes place, an attacker is able to display the edited data on the DDL app without any prevention.

Presumably, whatever moron designed this system figured that since it was encrypted with military-grade 256-bit AES encryption, and thus could never be modified, there would be no need to verify it. And one of the features they boasted was that it could run completely offline. Right. No verification needed.

And speaking of verification... One of the key “verification features” of the digital license is its “pull-to-refresh” functionality, which is used to ensure you are viewing the most current license information.

However, the Dvuln guys noticed that refreshing the application's driver license data only updates the QR code displayed on the license (and the QR code only contains the license holder’s name and under 18 age status). So if a fraudster had modified their license details and photo by decrypting, modifying and re-encrypting the data, this fraudulent data would remain visible on the screen even after the QR code, date and time had been refreshed and updated.

And, not surprisingly, in still another example of sloppiness, the license data is, indeed, exported in iPhone backups, making its modification trivial: As we know when a secured mobile device is jailbroken, it's reasonable to assume that any security features an application may have may be bypassed because an attacker has obtained root access to the devices services and storage. But, conversely, as long as a secured mobile device is **not** jailbroken, apps can reasonably secure their users against misuse and various types of client-side vulnerabilities.

However, in the case of the ServiceNSW application, the Digital Driver Licence data **is** included in device back-ups, which means that attackers or anyone wanting to commit fraud can obtain and modify their license details without ever needing to jailbreak their device.

There is no way that this system was **ever** reviewed by any competent security expert. These days we've got competent security experts coming out of our ears. Anybody who had any training in digital application security would immediately see what an embarrassment this thing is. And the publicly known problems with it are now three years old with rumors that these DDL's are being readily spoofed. Of course they are. So we have to ask ourselves... how did this happen? Was this thing created by some government bureaucrat's unemployed nephew? And why doesn't anyone appear to care? It seems quite likely that someone will be caring very soon thanks to the bright light that the Dvuln guys have aimed at this mess.

So what's the answer? How do we solve this problem? Since I've been called a competent security expert, I'll take a stab at it. How about this: all that's needed is a certificate. A standard X.509 format certificate. This has all been worked out already. Certificate fields can contain binary data — they already do, like a public key — so the certificate owner's photo can easily be contained within the certificate, as can dates of starting and ending validity and anything else that might be needed. The owner's legal name, their physical address, date of birth, and so on.

What makes a certificate special is that its entire contents is signed by a recognized and trusted authority. As we know, the process of signing is that the certificate's contents are hashed to create a digest of its contents. Then that hash is encrypted with the signer's private key. Anyone wishing to verify the certificate's authenticity simply creates their own hash of the certificate's contents, then uses the signer's public key to decrypt the hash that came with the certificate. If the new hash and the decrypted hash match, then we know that the certificate's contents cannot have been modified since it was signed.

So, to build a simple and practical system, an existing trusted root certificate authority, any existing certificate authority that's already trusted by the mobile iOS and Android platforms, like my favorite and chosen CA, DigiCert, issues an intermediate certificate to the New South Wales administration which is, itself, permitted to sign these special purpose end DDL certificates.

Any time someone out in the world, like a DDL license holder's DDL application, wishes to update and re-verify their license, that app sends a request to the government's server. The server pulls together all of the relevant data it has for the individual from its database, including their latest photo, their date of birth and their driver license's expiration date, builds a new certificate which also contains a current timestamp, and uses the private key it obtained from DigiCert to sign the resulting certificate. It then bundles that certificate with the intermediate certificate's public key, and sends the package back to the DDL application.

Since the bundle contains a short certificate chain whose intermediate certificate is already trusted by the root certificate store in the mobile device, and since that intermediate certificate can verify the DDL certificate, any iOS or Android device has everything it needs to verify the authenticity of the DDL end certificate. And there is no way for that DDL license certificate to be modified or tampered with in the field without breaking its signature.

In this system, there's no need for a PIN to encrypt the certificate's data. There's no need for any certificate encryption at all.

And as for that QR code that apparently only contains the user's name and age, that's also ridiculous if its data can be spoofed. If you want to have some sort of verification reader, then the QR code can simply contain the individual's driver's license number. The reader obtains the license number, makes a query to the government's server to obtain their signed DDL certificate, and displays the user's name, age, photo, and everything else.

And if the QR code system needs to work offline, which was apparently a feature of the current system, then the QR code can simply contain a signed subset of the user's information, like only their name and date of birth which is separately signed by the government's intermediate certificate. The QR code reader then scans the QR code which is, itself, a small certificate. It uses the government's intermediate certificate to verify the QR code certificate and can then trust that the data contained in that QR code certificate subset has not been tampered with.

Again, none of this is rocket science. As I've said many times, we now have an incredible toolkit of technology components that can be applied individually and collectively to solve **any** of these sorts of problems. Nothing needs to be invented. And the beauty of this system is that all of the well-tested and bullet-proof crypto libraries already exist. In fact, the iOS and Android platforms already contain all of the required crypto machinery APIs. None of that needs to be created.

Hopefully, when this ridiculous New South Wales digital driver's license disaster fiasco finally does come to light, the existing ridiculous system will **not** be salvaged, but it will be entirely scrapped and replaced with a simple and bullet-proof system like the one I've just described.

The latest Microsoft Office 0-day remote code execution vulnerability

We have a new, head-buried-in-the-sand, quite pervasive Microsoft Office 0-day remote code execution vulnerability which is now currently being used in attacks.

By far the best researcher on this has been Kevin Beaumont who even tracked down a Bachelor's thesis authored by Benjamin Altpeter on August 1st, 2020, so nearly two years ago. On page 29 of his thesis, Benjamin writes:

Windows includes the ***ms-msdt:*** *protocol that opens the Microsoft Support Diagnostic Tool which provides the troubleshooting wizard to diagnose Wi-Fi and audio problems and the like. This protocol directly passes the string it is given to the* ***msdt.exe*** *program. The attacker now needs to find an included wizard that allows the execution of arbitrary programs, preferably even remote ones. The program compatibility wizard fits this description. Luckily for the attacker, all user input can also be prefilled from the command line.*

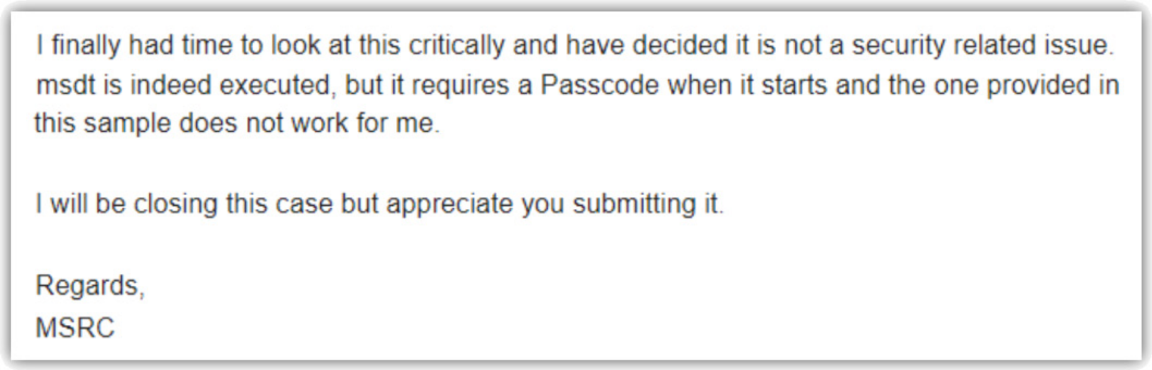
At that point I opened my own command-line window on my Windows machine, entered “msdt” and hit enter... and sure enough, up popped a “Microsoft Support Diagnostic Tool” dialog. So this MSDT thing is alive and well and living in all of our machines right now.

Okay, so back to Kevin Beaumont: Kevin's name for this exploit vector has stuck. It's now semi-officially known as “Follina” because Kevin spotted a reference to 0438 in the sample exploit file and 0438 is the area code of the city of Follina, Italy.

So Benjamin's Bachelor's thesis was nearly two years ago. The reference in it was obscure and we'll never know whether someone saw it and recognized its significance or independently invented an attack. Given the nearly two-year interval, I'm inclined to think that this was an independent discovery.

But either way, last month on April 12th, the leader of Shadowchasing1, an advanced persistent threat (APT) hunting group reported the active exploitation of this vulnerability in the wild to Microsoft’s MSRC their “Microsoft Security Response Center.” The Shadowchasing1 report provided a copy of the in the wild, real world Microsoft Office document exploit targeting Russia, themed as a Russian job interview.

Nine days later, on April 21st, Microsoft’s MSRC blew it off and closed the ticket saying that it was not a security related issue:


I finally had time to look at this critically and have decided it is not a security related issue. msdt is indeed executed, but it requires a Passcode when it starts and the one provided in this sample does not work for me.

I will be closing this case but appreciate you submitting it.

Regards,
MSRC

Now, I’m not sure what the phrase “looking at this critically” means, exactly. But apparently it doesn’t mean “closely examined and worked to understand what the underlying problem might be.” And, you know, this might represent another sample of the pattern that seems to be emerging, where Microsoft increasingly seems to be needing the external security research community to solve all of its problems for it, and hand it everything on a silver platter.

Apparently saying: “Hey, have you considered that allowing Word’s remote template feature to retrieve an HTML file from a remote web server, which in turn uses the ms-msdt:// URI scheme to load some code and execute some PowerShell — even when when Office Macros have been disabled and even able to bypass Protected View by giving the document an RTF extension?” is not sufficient to get Microsoft’s attention.

Now, in fairness, the provided exploit sample apparently did not work instantly for the overworked MSRC guy. And we don’t know how many false-negative reports the MSRC guys might be fielding every day. Perhaps they face a constant deluge of bogus reports.

But we do know about economics and we do know about incentives. So we know that if this sort of behavior is never met with consequences, it will continue and even be encouraged. And Microsoft has arranged to completely insulate themselves from any consequences. So all we can ever do is hope for the best.

Since we're apparently on our own, here, what do we know? Kevin reports that the vulnerability has been proven to work on Office 2013, 2016, 2019, 2021, Office ProPlus and Office 365. It also applies to Windows itself because it can be called from .lnk link files. So there are two different issues: Office itself using the ms-msdt: scheme protocol while allowing loading unfiltered HTML Word templates and Outlook links, and the fact that the MSDT executable allows code execution. All flavors of Windows Defender also completely misses detection of this, which is not surprising given that Microsoft decided last month that this was not a security issue.

That finally changed yesterday with Microsoft issuing a workaround in the form of a registry script which simply deletes the ms-msdt protocol handler from the registry:

1. Run Command Prompt as Administrator.
2. Execute the command "reg delete HKEY_CLASSES_ROOT\ms-msdt /f".

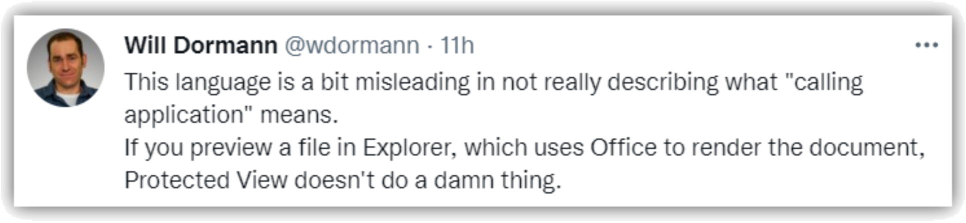
Kevin wonders out loud how this might evolve, saying:

We'll see. Microsoft are going to need to patch it across all the different product offerings, and security vendors will need robust detection and blocking. Microsoft will probably point towards Protected View, however Protected View also applies by default to all macros, and Office macro malware is most definitely a major problem regardless.

Then, in an update to Kevin's original posting, he added:

Microsoft have indeed pointed to Protected View, saying it "prevents" the attack. I think this is stretching the truth — for example, if the document is a .RTF file and is opened by Preview in Explorer, Protected View doesn't apply and it becomes a zero click exploit. Microsoft knows this, they just aren't mentioning it to customers.

Kevin then provides a tweet from CERT/CC's Will Dormann which agrees:


Will Dormann @wdormann · 11h
This language is a bit misleading in not really describing what "calling application" means.
If you preview a file in Explorer, which uses Office to render the document, Protected View doesn't do a damn thing.

The very latest is that Microsoft has finally woken to this threat, a CVE has been assigned and Windows Defender's detection signatures have been updated. But the underlying trouble is that the use of MS Office protocol is extensive and pervasive, so it cannot be shutdown without breaking all sorts of other things that depend upon it.

It’s a bit like last year’s printer spooling Catch-22. It’s another of those problems that isn’t really a bug that can be patched, because it’s the abuse of a deliberately designed-in feature. And this what results from systems that become too complex. It begins to be impossible to understand and anticipate every possible interaction among components when there are just so many different components.

GhostTouch

Get a load of this one. This is not yet a real threat due to the potential attack's very short range which is currently around 40mm, or just over an inch and a half. But as we know, “impractical” is how many eventually practical attacks begin. Here's the Abstract of the 17-page exploit research paper describing the new and very clever “GhostTouch” attack which was invented by a team of Chinese and German researchers. They explain...

Capacitive touchscreens have become the primary human-machine interface for personal devices such as smartphones and tablets. In this paper, we present GhostTouch, the first active contactless attack against capacitive touchscreens. GhostTouch uses electromagnetic interference (EMI) to inject fake touch points into a touchscreen without the need to physically touch it. By tuning the parameters of the electromagnetic signal and adjusting the antenna, we can inject two types of basic touch events, taps and swipes, into targeted locations of the touchscreen and control them to manipulate the underlying device. We successfully launch the GhostTouch attacks on nine smartphone models. We can inject targeted taps continuously with a standard deviation of as low as 14.6 × 19.2 pixels from the target area, a delay of less than 0.5s and a distance of up to 40mm. We show the real-world impact of the GhostTouch attacks in a few proof-of-concept scenarios, including answering an eavesdropping phone call, pressing the button, swiping up to unlock, and entering a password. Finally, we discuss potential hardware and software countermeasures to mitigate the attack.

A capacitive touchscreen is a surprisingly sophisticated and highly sensitive device and the technology is known to be quite susceptible to local environmental noise. It's easy to make a touchscreen malfunction when it's placed near a source of electromagnetic interference. The switching regulators used in some phone chargers are known to generate so much short-range electromagnetic interference that nearby touchscreens will not function. So what these clever researchers have done is to turn electromagnetic interference into electromagnetic touch signals to deliberately spoof the signals generated by legitimate touch events.

At this point it’s only of academic interest. But we should not be surprised to learn of it being applied in some way in the future.

Vodafone’s new TrustPiD

There's something new that might be coming known as TrustPiD. If it ever actually happens — and Vodafone is reportedly now pilot testing their new TrustPiD system in Germany — the technology creates a persistent, static, super-cookie which, being somehow injected into a user’s communications at the carrier-level by the cellular carrier, cannot be seen, managed or blocked by end users.

The only reason I can see for any cellular carrier to be doing something this clearly privacy invasive is that they've decided that it's more important for them to get in on the Internet advertising revenue boom by arranging to monetize their customer's "anonymous" (and that's in quotes) online identities.

Here's what the TrustPiD website says under the tired banner of "Keeping the Internet free":

> Consumers appreciate the idea of a 'free' Internet, but this comes with a trade-off: publishers need a sustainable revenue model, meaning that it becomes essential to add subscription paywalls or rely on advertising to maintain free access to high quality content.
>
> With a growing trend of digital information shared in the ecosystem, consumers' concerns about privacy and the amount of information passed into the free Internet have been raised.
>
> TrustPid is a technology solution that enables consumers to enjoy free content and the benefits of the open Internet whilst retaining control over their privacy.
>
> TrustPid is a secure, unique digital 'token' generated by assigning random numbers to you which reduces the risk of you being directly identified whilst still enabling advertisers and publishers to provide you with a personalized experience across their sites with your consent. You can find more information on how we generate and manage your TrustPid in the Privacy Notice.
>
> Your consent for TrustPid is collected by advertisers and publishers via their Consent Management Platform (Cookie banner) when you visit their sites. The consent will apply only to those websites.
>
> TrustPid service gives consumers complete control over how their TrustPid is used by enabling them to manage their consent via a central "Privacy Portal", at any time.
>
> In the Privacy Portal, consumers are able to track which advertisers or publishers they have allowed to provide them with personalized online marketing in their websites based on their TrustPid in one central place; the possibility to withdraw consent at single website level and the ability to turn off the TrustPid service entirely, preventing any further use of the token.
>
> If you want to understand more how the Privacy Portal works or if you want to manage your consent, please visit the dedicated Privacy Portal page.

Okay. So I have several questions here. Aside from the fact that this will assuredly be an Opt-Out service where all users will be, by default, opted-in, and thus enabling tracking without their consent (and we need to know whether this system will respect the user's GPC – Global Privacy Control – setting.)

As I said, aside from that, HOW exactly can identifying tags be injected at the carrier level into encrypted HTTPS web sessions? This is really a concern. The only way I can see that this can be done in 2022 is by having the carrier, in this case Vodafone, actively intercepting and proxying all of their customer’s encrypted TLS connections. We saw this years before when some carriers were offering caching and data compression services for their customers. But that was back in the pre-HTTPS-Everywhere days, where most connections were still unencrypted. But in 2022, nothing is unencrypted anymore. A carrier can see the IP addresses that their users are connecting to. But IP addresses no longer reflect the web site properties behind them, since SNI — Server Name Indication — is now being heavily used during TLS handshaking to identify the domain being connected to and thus the TLS certificate to send to the client. This allows many websites to reside at the same IP.

So this means that a persistent identifier is either somehow injected into the user’s connections by their smartphone’s browser before entering their encrypted TLS connection, or the user’s smartphone must accept a Vodafone CA-style root certificate into its root store to allow Vodafone to function as a sanctioned, active, persistent cookie injecting man-in-the-middle.

Neither of these seem likely, possible or even remotely feasible, in 2022. So I’m stumped. Without being on the inside, Vodafone cannot see who their customers are connecting to. And cellular carriers are explicitly and deliberately on the outside.

Oh, and T-Mobile's parent company, Deutsche Telekom, is also reportedly testing this same carrier-grade super-tracking cookie technology. I’ve followed every link I could find looking for any technical documentation. Since I don’t read German my digging is somewhat limited. Maybe I’m missing something obvious. But even if there’s some way to pull this off technically, why the hell are our carriers getting in on the “identifying their customers to advertisers” game?

This monster needs to be quickly strangled in its crib.

Closing The Loop

MikeO / @emohsee

Hi Steve (@SGgrc) & Leo (@leolaporte). Listed to SN 872 and you said there are no good ad blockers for macOS/iOS. I’ve been using 1Blocker for years and it works great with Safari on all of Apple’s platforms: 1blocker.com

Bill Zegarski / @billzegarski

@SGgrc heard the conversation on ad blockers for Safari. @1BlockerApp, which you recommended and I love on iOS, also has a Mac version. Might want to look at that. Best part is that since it is a universal app, if you purchase on one platform you get both!

george palfi / @PalfiGeorge

Just heard your discussion of Quantum computing and cryptography. Perhaps we can adapt new crypto for Quantum computers. But, Bitcoin, Ethereum and many cryptocurrencies are based on very old crypto rules. And, I don’t believe the algorithms can be changed. So, people who want to keep their wealth protected for a long time will be at risk as Quantum computing evolves. Better to stick to gold and silver which have worked for thousands of years.

[That's a good point, George. But the good news is that Cryptocurrencies, at least Bitcoin, use a 256-bit hash which, as we know, is for now at least believed to be strongly quantum resistant.]

r4nd0m / @r4nd0mpr1m35

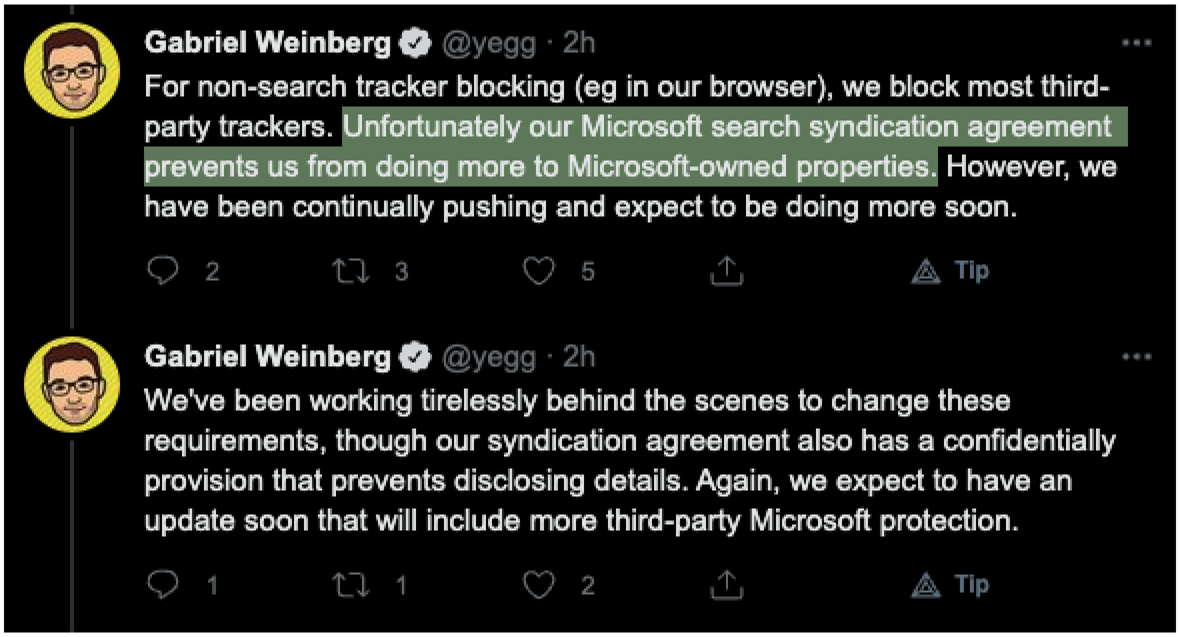
Hey @SGgrc I was thinking about the layer-1 attack against the unlock function of the various apps using BLE and it occurred to me that a prompt, in the app, at unlock time should be a feature they add to help mitigate the demonstrated weakness. Of course the UX will suffer but this should defeat the attack.

Bryant McDiarmid / @bryantmcdiarmid

Hey Steve, quick question. If I encrypt a file with a 256 bit encryption three times with three different passwords, what is the resulting bit strength? Is it 256 plus 256 plus 256? Or 256 times 256 times 256?

DuckDuckGone?

The winner of this week's most tweeted news-of-concern was the widely reported surprise that the explicitly privacy centric and privacy protecting DuckDuckGo enterprise had struck a previously secret backroom deal with Microsoft to enable user tracking from Microsoft owned domains including bing.com and linkedin.com when using the so-called **DuckDuckGo Privacy Browser**. It was further revealed by DuckDuckGo's founder Gabriel Weinberg that DuckDuckGo's non-disclosure agreement with Microsoft prevented them from any further disclosure of their agreement's terms.



Gabriel Weinberg @yegg · 2h

For non-search tracker blocking (eg in our browser), we block most third-party trackers. Unfortunately our Microsoft search syndication agreement prevents us from doing more to Microsoft-owned properties. However, we have been continually pushing and expect to be doing more soon.

Gabriel Weinberg @yegg · 2h

We've been working tirelessly behind the scenes to change these requirements, though our syndication agreement also has a confidentially provision that prevents disclosing details. Again, we expect to have an update soon that will include more third-party Microsoft protection.

Okay. So let's back up a bit and see what happened...

- TechCrunch's headline: "DDG has a tracker blocking carve-out linked to Microsoft contract"
- BleepingComputer: "DuckDuckGo browser allows Microsoft trackers due to search agreement"
- 9to5mac: "DuckDuckGo caught giving Microsoft permission for trackers despite strong privacy reputation"
- Android Police: "DuckDuckGo's supposedly private browser caught permitting ad tracking"

The first thing for us to be clear about is that all of this refers to the use of DuckDuckGo's own browser, not to the use of the DuckDuckGo search engine. On the other hand, anyone could be forgiven for missing this distinction. When you go to DuckDuckGo's homepage you're greeted with the bold headline: *"Tired of being tracked online? We can help."* Then below that are three big topics: "Privacy for Chrome", "Private Search Engine" and "Privacy Browser App" And explaining their privacy browser app, they say: *"Our private browser for mobile comes equipped with our search engine, tracker blocker, encryption enforcer, and more. Available on iOS & Android."* — and as we've previously mentioned, a version for Windows is planned.

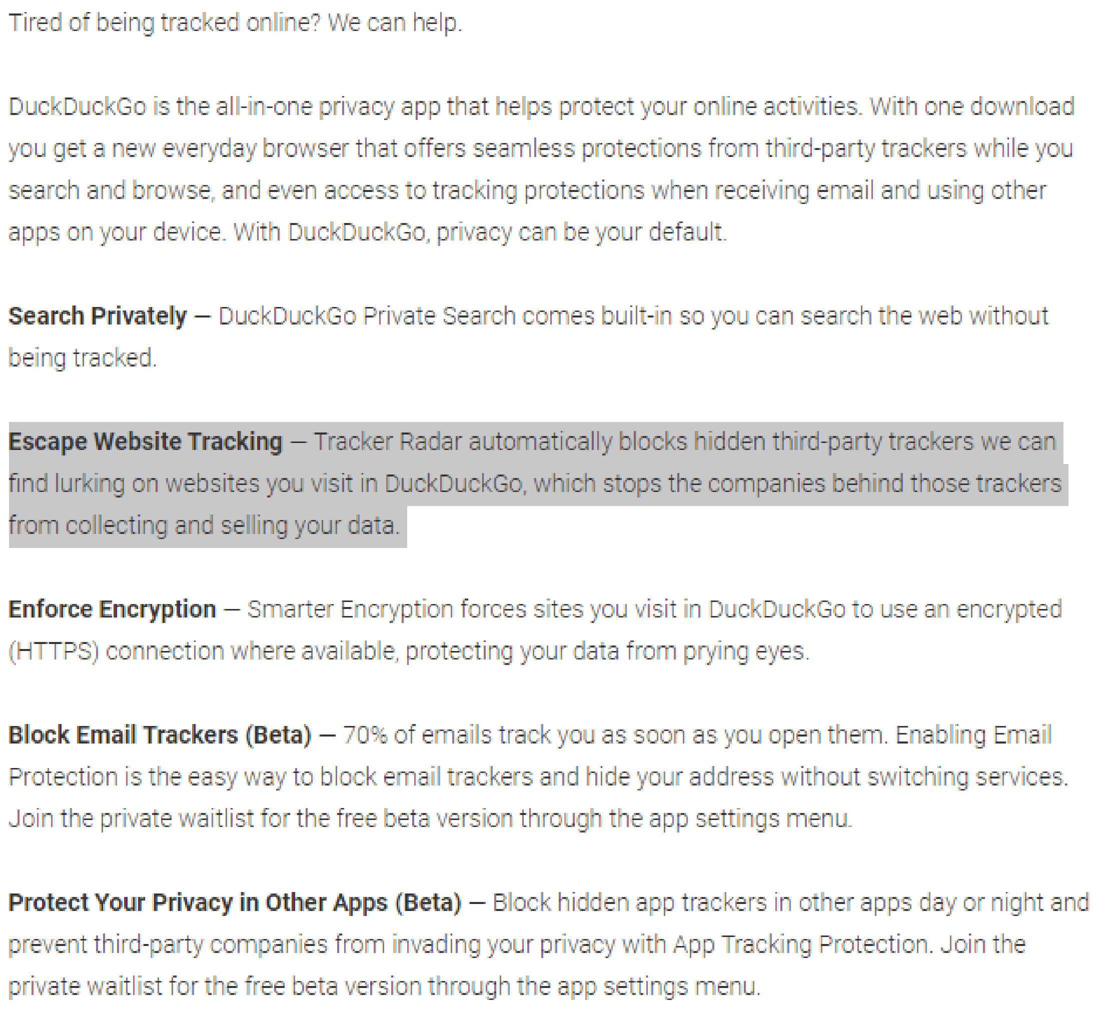
All of this started when security & privacy researcher Zach Edwards took the time to audit the data flows to and from one of DuckDuckGo's mobile platform browsers. Early last week he tweeted:

> *"Sometimes you find something so disturbing during an audit, you've gotta check / recheck because you assume that *something* must be broken in the test. But I'm confident now. The new @DuckDuckGo browsers for iOS & Android don't block Microsoft data flows, for LinkedIn or Bing."*

This is mostly a story about expectations. Zach was so surprised and disturbed by what he saw specifically because it wasn't what he ever expected to find from DuckDuckGo. His next tweet:

> DuckDuckGo has browser extensions & their own browsers for iOS / Android @ https://duckduckgo.com/app
> iOS @ https://apps.apple.com/us/app/duckduckgo-privacy-browser/id663592361
> Android @ https://play.google.com/store/apps/details?id=com.duckduckgo.mobile.android
> Both versions of the DDG browser claim to use tools which "automatically blocks hidden third-party trackers" ??

And to back that up, Zach attached a screenshot of those clear statements:



Tired of being tracked online? We can help.

DuckDuckGo is the all-in-one privacy app that helps protect your online activities. With one download you get a new everyday browser that offers seamless protections from third-party trackers while you search and browse, and even access to tracking protections when receiving email and using other apps on your device. With DuckDuckGo, privacy can be your default.

Search Privately — DuckDuckGo Private Search comes built-in so you can search the web without being tracked.

Escape Website Tracking — Tracker Radar automatically blocks hidden third-party trackers we can find lurking on websites you visit in DuckDuckGo, which stops the companies behind those trackers from collecting and selling your data.

Enforce Encryption — Smarter Encryption forces sites you visit in DuckDuckGo to use an encrypted (HTTPS) connection where available, protecting your data from prying eyes.

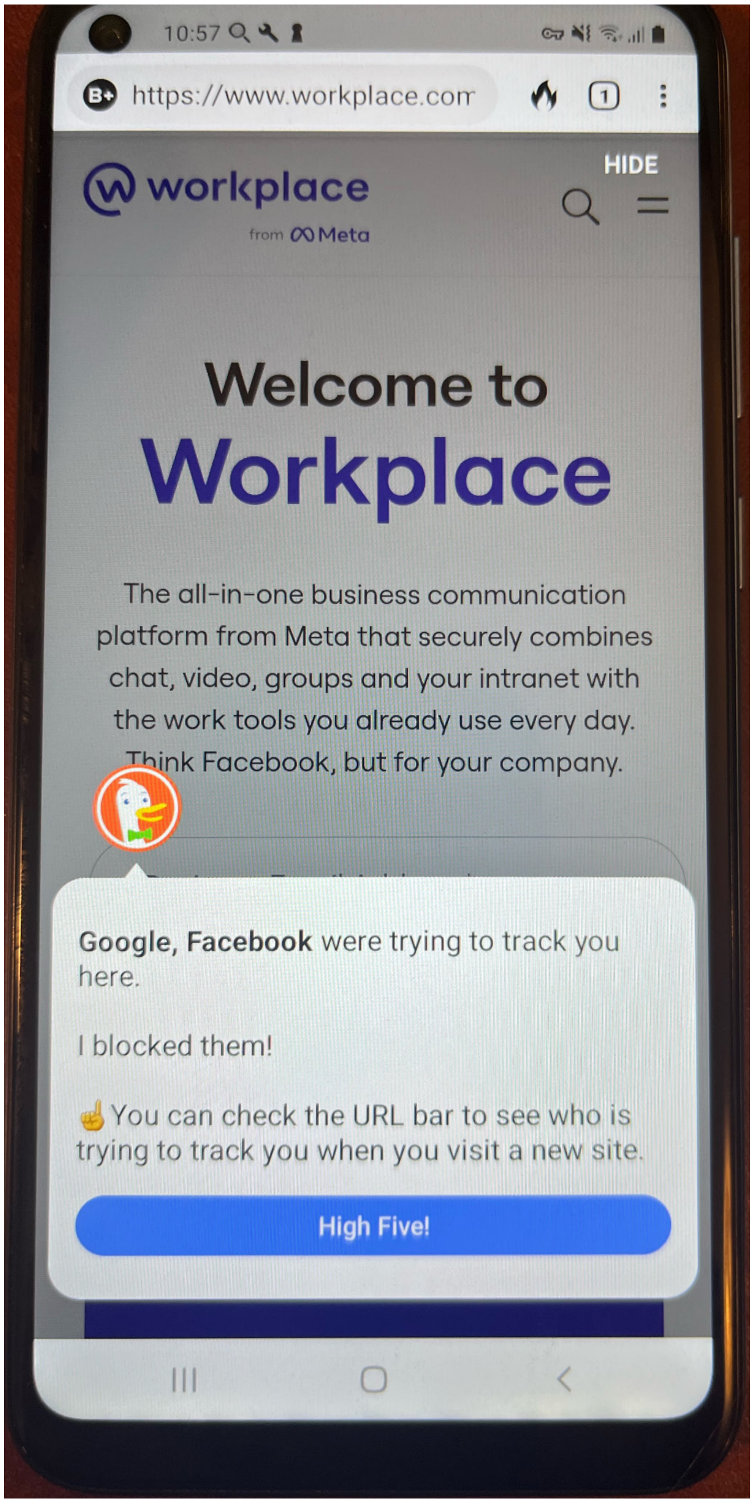
Block Email Trackers (Beta) — 70% of emails track you as soon as you open them. Enabling Email Protection is the easy way to block email trackers and hide your address without switching services. Join the private waitlist for the free beta version through the app settings menu.

Protect Your Privacy in Other Apps (Beta) — Block hidden app trackers in other apps day or night and prevent third-party companies from invading your privacy with App Tracking Protection. Join the private waitlist for the free beta version through the app settings menu.

It doesn't say "Except for Microsoft's advertising trackers which our contract with them prevents us from blocking or disclosing." Zach's next tweet:

*I tested the DuckDuckGo so-called private browser for both iOS and Android, yet *neither version* blocked data transfers to Microsoft's Linkedin + Bing ads while viewing Facebook's workplace[.]com homepage.*

Look at DDG bragging about stopping Facebook on Workplace, no [mention of] Microsoft..:


10:57

https://www.workplace.com

HIDE

workplace from Meta

Welcome to Workplace

The all-in-one business communication platform from Meta that securely combines chat, video, groups and your intranet with the work tools you already use every day. Think Facebook, but for your company.

Google, Facebook were trying to track you here.

I blocked them!

☝ You can check the URL bar to see who is trying to track you when you visit a new site.

High Five!

And Zach posted a photo showing his Android phone at Facebook's workplace.com site with a DuckDuckGo pop-up bragging about the wonderful job it's doing, saying:

Google, Facebook were trying to track you here. I blocked them! You can check the URL bar to see who is trying to track you when you visit a new site.

And then there's a big "High Five!" button which the happy user can presumably press.

The fact that this browser specifically singles out other tracking properties while silently permitting Microsoft-own domains to track, feels at best disingenuous.

In an attempt to clarify the mess that arose from this, Gabriel appeared to attempt to explain what's going on over on Reddit.

You'll hear for yourself in a moment why I'm wording this as provisionally as I am, because although I've read what Gabriel wrote several times slowly and carefully — and it sort of sounds like he's explaining something — I still have no idea what he said. I assume he knows, but he sure didn't communicate it.

Here's what Gabriel wrote:

Hi, I'm the CEO & Founder of DuckDuckGo. To be clear (since I already see confusion in the comments), when you load our search results, you are anonymous, including ads. Also on 3rd-party websites we actually do block Microsoft 3rd-party cookies in our browsers plus more protections including fingerprinting protection. That is, this article is not about our search engine, but about our browsers -- we have browsers (really all-in-one privacy apps) for iOS, Android, and now Mac (in beta).

When most other browsers on the market talk about tracking protection they are usually referring to 3rd-party cookie protection and fingerprinting protection, and our browsers impose these same restrictions on all third-party tracking scripts, including those from Microsoft. We also have a lot of other above-and-beyond web protections that also apply to Microsoft scripts (and everyone else), e.g., Global Privacy Control, first-party cookie expiration, referrer header trimming, new cookie consent handling (in our Mac beta), fire button (one-click) data clearing, and more.

*What this article is talking about specifically is another above-and-beyond protection that most browsers don't even attempt to do for web protection— stopping third-party tracking scripts from even loading on third-party websites -- because this can easily cause websites to break. But we've taken on that challenge because it makes for better privacy, and faster downloads -- we wrote a blog post about it here. Because we're doing this above-and-beyond protection where we can, and offer many other unique protections (e.g., Google AMP/FLEDGE/Topics protection, automatic HTTPS upgrading, tracking protection for *other* apps in Android, email protection to block trackers for emails sent to your regular inbox, etc.), users get way more privacy protection with our app than they would using other browsers. Our goal has always been to provide the most privacy we can in one download.*

The issue at hand is, while most of our protections like 3rd-party cookie blocking apply to Microsoft scripts on 3rd-party sites (again, this is off of DuckDuckGo,com, i.e., not related to search), we are currently contractually restricted by Microsoft from completely stopping them from loading (the one above-and-beyond protection explained in the last paragraph) on 3rd party sites. We still restrict them though (e.g., no 3rd party cookies allowed). The original example was Workplace.com loading a LinkedIn.com script. Nevertheless, we have been and are working with Microsoft as we speak to reduce or remove this limited restriction.

I understand this is all rather confusing because it is a search syndication contract that is preventing us from doing a non-search thing. That's because our product is a bundle of multiple privacy protections, and this is a distribution requirement imposed on us as part of the search syndication agreement that helps us privately use some Bing results to provide you with better private search results overall. While a lot of what you see on our results page privately incorporates content from other sources, including our own indexes (e.g., Wikipedia, Local listings, Sports, etc.), we source most of our traditional links and images privately from Bing (though because of other search technology our link and image results still may look different). Really only two companies (Google and Microsoft) have a high-quality global web link index (because I believe it costs upwards of a billion dollars a year to do), and so literally every other global search engine needs to bootstrap with one or both of them to provide a mainstream search product. The same is true for maps btw -- only the biggest companies can similarly afford to put satellites up and send ground cars to take streetview pictures of every neighborhood.

Anyway, I hope this provides some helpful context. Taking a step back, I know our product is not perfect and will never be. Nothing can provide 100% protection. And we face many constraints: platform constraints (we can't offer all protections on every platform do to limited

APIs or other restrictions), limited contractual constraints (like in this case), breakage constraints (blocking some things totally breaks web experiences), and of course the evolving tracking arms race that we constantly work to keep ahead of. That's why we have always been extremely careful to never promise anonymity when browsing outside our search engine, because that frankly isn't possible. We're also working on updates to our app store descriptions to make this more clear. Holistically though I believe what we offer is the best thing out there for mainstream users who want simple privacy protection without breaking things, and that is our product vision.

So here's what I **think** Gabriel said:

We want to provide a robustly privacy-enforcing search engine service. But to do that we first need to have a search engine service, and no one other than Google and Microsoft can do that on their own. So if we want to provide search, we need to purchase access to a big search index. And we chose Microsoft's. Because we're able to sanitize and purify the link results we provide, we're able to offer tracking-free search... and we do.

But completely aside from and separate from web search, we also wanted to provide a privacy-enhancing web browser. We wanted to do this because offering clean search results is only part of the problem. A privacy-centric web browser can do so much more. And one of the "so much more" things a browser can do is not only block cookies being set and read by 3rd-parties, but also block their 3rd-party JavaScript code from ever being run in the user's browser. And we're able to do that, to whatever degree we can, for everyone other than when a Microsoft property is that 3rd party. In that case we must allow their JavaScript to run. This is not because we want to make an exception for Microsoft. It's because the completely unrelated agreement we have, which allows us to have access to their Bing search index, explicitly requires that we not block the execution of their scripts in our browser.

Because this has all come to light now, we're planning to amend our DuckDuckGo browser description pages to say something about this.

So, I think their heart's in the right place. They would not be making an exception to allow Microsoft's domains to run 3rd-party scripts if their search index syndication contract did not require it. They're not happy about that, either. But it's the price they pay for being able to offer their flagship DuckDuckGo tracking-free search service. And the fact is that blocking all other 3rd-party scripts and all 3rd-party cookies, and doing everything else possible, really does create a very compelling privacy-centric browser.