Security Now! #856 - 02-01-22

The “Topics” API

This week on Security Now!

This is another of those weeks where we're going to go deeper into fewer topics rather than broader across more topics, with Google's newly announced and explained “Topics” API of course being our title story. So we'll start by looking at “PwnKit” which is a startling and long standing local privilege escalation vulnerability which has existed in every distribution of Linux since May of 2009. It's a MUST PATCH for Linux systems. We'll then look at another of the blessedly few Log4j exploits which is actually happening, update on two new Zerodium limited-time bounty “offers” and at a new means for fingerprinting web browsers. I have a totally random bit of miscellany to share in the form of a tip, a SpinRite update and some closing the loop feedback from our terrific listeners. Then we'll wrap up by taking a really interesting deep dive into Google's new ad-targeting “Topics” API.

The absolutely cleanest “Binary Search”
I’ve ever written in x86 Intel assembly:

```
;
;;╔═══════════════════════════════════════════════════════════════════════════╗
;;║                                BINARY SEARCH                              ║
;;╟───────────────────────────────────────────────────────────────────────────╢
;;║     Rather than bouncing a probe, this uses the "Binary Chop" method      ║
;;║     of narrowing the range by moving the left and right walls inward.     ║
;;╟───────────────────────────────────────────────────────────────────────────╢
;;║     Given a target value, an Array offset and the Array's length,         ║
;;║     this returns the index of the located value within the array.         ║
;;╚═══════════════════════════════════════════════════════════════════════════╝
BinarySearch    PROC USES ebx, Target:DWORD, Array:LPVOID, Len:DWORD
;-------------------------------------------------------------------------------
TES = 8 ; Table Entry Size - sets the size of individual table entries
        ; -TES forms a mask for entries of "TES" size
;-------------------------------------------------------------------------------
        mov     eax, Array              ; EAX has the lower edge index
        mov     ecx, Len
        lea     ecx, [eax+ecx-TES]      ; ECX has the upper edge index
        mov     ebx, Target             ; EBX has our search target

        .WHILE (eax <= ecx)             ; so long as the edges haven't crossed
                lea     edx, [eax+ecx]  ; set the test midway between
                halve   edx             ; get midpoint estimate
                and     edx, -TES       ; round it down to entry boundary

                .IF ([edx] == ebx)          ; compare this test to our target
                        mov     eax, edx    ; we FOUND our target
                        jmp     Exit        ; so were done!
                .ELSEIF (carry?)
                        mov     eax, edx    ; midpoint was LOWER than target
                        add     eax, TES    ; so move the lower edge UP past it
                .ELSE
                        mov     ecx, edx    ; midpoint was HIGHER then target
                        sub     ecx, TES    ; so more the upper edge DOWN past
                .ENDIF
        .ENDW
        zero    eax     ; the upper and lower edges crossed, so not found
Exit:   ret             ; return a pointer to the target, or NULL if none
;-------------------------------------------------------------------------------
BinarySearch    ENDP
```

0-Day Watch

Apple eliminates 0-days from iOS and macOS

Wednesday, Apple released iOS 15.3 and macOS Monterey 12.2 which included fixes for two 0-day vulnerabilities, one publicly disclosed and the other exploited in the wild by attackers to break into iPhones, iPads and Macs. These are the first 0-days patched by Apple in 2022, though as we know, Apple was patching a continuous stream of 0-day bugs throughout 2021 which were being exploited in targeted attacks, often to install the Pegasus spyware on the iPhones of journalists, activists, and politicians.

The 0-day, was tracked as CVE-2022-22587, stemmed from a memory corruption issue in the IOMobileFrameBuffer component that could be abused by a malicious application to execute arbitrary code with kernel privileges and it's worth noting that this is the 3rd 0-day discovered in IOMobileFrameBuffer within the past 6 months. Last year Apple fixed two others. Also in December, Apple resolved four other weaknesses in the kernel extension that's used to manage the screen's framebuffer.

As for being a 0-day, Apple said that it's "aware of a report that this issue may have been actively exploited" and added that it had addressed the issue with improved input validation. Apple did not reveal anything about the nature of the attacks, how widespread they are, or the identities of the threat actors exploiting them.

A privacy-defeating bug in Safari was also eliminated. That one was the result of bugs in the implementation of the IndexedDB API (CVE-2022-22594), which could be abused by malicious websites to track users' online activity in Safari and reveal their identity. That one had been publicized by researchers at FingerprintJS as a WebKit flaw affecting macOS, iOS and iPadOS. Its exploitation allows a snooping website to discover information about other tabs a user might have open.

That bug is a cross-origin policy violation in the IndexDB API which is a JavaScript API provided by web browsers to manage a NoSQL database of JSON objects. Apple closed this loophole by improving the API's input validation.

By design, a web browser only permits scripts on one web page to access data on a second web page only if both pages have the same domain name origin. Without a browser's cross-origin policy protections, a malicious site or script in an advertisement might be able to obtain access to data contained in other tabs the user might have open in the browser which might include online banking sessions, emails, healthcare portal data and other sensitive information. So that Safari problem is foreclosed. And also fixed were:

- CVE-2022-22584 – A memory corruption issue in ColorSync that may lead to arbitrary code execution when processing a malicious crafted file
- CVE-2022-22578 – A logic issue in Crash Reporter that could allow a malicious application to gain root privileges
- CVE-2022-22585 – A path validation issue in iCloud that could be exploited by a rogue application to access a user's files
- CVE-2022-22591 – A memory corruption issue in Intel Graphics Driver that could be abused by a malicious application to execute arbitrary code with kernel privileges

- CVE-2022-22593 – A buffer overflow issue in Kernel that could be abused by a malicious application to execute arbitrary code with kernel privileges
- CVE-2022-22590 – A use-after-free issue in WebKit that may lead to arbitrary code execution when processing maliciously crafted web content

So, overall a useful collection of updates from Apple. When I checked my iPhone this morning it was still back on the previous release, 15.2.1. So it's not crucial, but you might want to check to see whether you're at the latest and, if not, give your device a bit of a nudge.

Security News

PwnKit

If Google's big reveal last week of their proposed "Topics" API hadn't seemed like such a potentially significant event for the industry, Qualys' disclosure, that same day, of an absolutely pervasive, longstanding and readily exploited local privilege escalation vulnerability affecting every major Linux distribution — being referred to as an attacker's dream come true — would certainly have been our title story for the week. Fortunately, we have the time to discuss both and this one is really interesting...

So... the Qualys Research Team discovered a memory corruption vulnerability in polkit's pkexec, which is an SUID-root program installed, by default, in every major Linux distribution. An SUID-root program is one which has its special "SUID" permission bit set which causes the operating system to run it under the permissions of its owner rather than the user who is invoking it. Since pkexec's owner is the root account, this give it a lot of power. And it turns out that there's an easily exploitable vulnerability in this program which allows any unprivileged user to gain full root privileges on a vulnerable host by exploiting this vulnerability in its default configuration with 100% reliability.

SANS Security Research wrote: "We expect that the exploit will become public soon and that attackers will start exploiting it – this is especially dangerous for any multi-user system that allows shell access to users." And SANS' expectations were realized less than three hours after Qualys published the technical details for what's being called PwnKit. I have a link to the C code for a reliable exploit at the end of this story.

The US's National Security Agency (you know, the NSA) Cybersecurity Director Rob Joyce noted on Twitter that the bug "has me concerned. Easy and reliable privilege escalation preinstalled on every major Linux distribution. Patch ASAP or use the simple chmod 0755 /usr/bin/pkexec mitigation. There are working POCs in the wild."

Polkit, which was formerly known as "PolicyKit", is a component used for controlling system-wide privileges in Unix-like operating systems, most popularly, Linux. This package is used for controlling system-wide privileges. The pkexec tool, which is a command line utility, is used to define which authorized user can execute a program as another user... and that utility has a critical flaw.

Qualys security researchers re-discovered the vulnerability, developed an exploit, and obtained full root privileges on default installations of Ubuntu, Debian, Fedora, and CentOS. Other, if not

all, Linux distros are likely vulnerable and probably exploitable. Most sobering is that this vulnerability has been hiding in plain sight for more than 12 years. It affects all versions of pkexec since its premiere release in May of 2009 when the mis-written pkexec command was first added to the PolicyKit. I mentioned that Qualys "re-discovered" the vulnerability. I'll explain about that after we talk about what it is.

As soon as Qualys' team had confirmed the vulnerability, they responsibly disclosed the glaring flaw back on November 18th and coordinated with all open-source distributions to fix and announce the vulnerability.

Qualys' disclosure (I have provided a link in the show notes) provides the details of the coding error that has always been present in polkit's pkexec command. And, frankly, it's a little shocking.

https://blog.qualys.com/vulnerabilities-threat-research/2022/01/25/pwnkit-local-privilege-escalation-vulnerability-discovered-in-polkits-pkexec-cve-2021-4034

In the C language, a command-line program receives two parameters from the operating system which is launching the program. They're commonly known as 'argc' and 'argv'. 'c' is short for count and 'v' is short for vector — as in a one-dimensional vector array. So argc is an integer count of the command-line parameters being passed to the program, and argv is a pointer to a vector array of pointers to strings. That sounds overly complicated when it's stated like that. But it just means that argv points to a list where the various command-line argument strings can be found, and argc tells us how long that list is — how many string parameters are in the list.

The problem is, launching a program from a command line is only one of several ways for a UNIX-like operating system to run a program. It's entirely possible for the operating system to launch a program itself. And in that case there's no actual command line and the operating system can determine what parameters it wishes to provide to the spawned program, if any. In other words, that list of parameters can be empty. In that case, 'argc'—the count of parameters —would be 0 and any properly designed parameter processing logic would know to skip that phase of the program's startup.

So guess what mistake the original author of pkexec made back in May of 2009? He completely failed to take into account the possibility that pkexec might be started **without** any parameters. He never checks to see whether argc is zero. His code mistakenly assumes that parameters will always be present, so it doesn't check. It just jumps right into dealing with the items in the nonexistent list of parameters... And it turns out that mistake can be weaponized and when it is, it's 100% reliable and even cross-architecture, also working on ARM64 systems which was confirmed by the CERT Coordination Center's vulnerability analyst Will Dormann.

RedHat described it this way: *"The current version of pkexec doesn't handle the calling parameters count correctly and ends up trying to execute environment variables as commands. An attacker can leverage this by crafting environment variables in such a way it'll induce pkexec to execute arbitrary code. When successfully executed, the attack can cause a local privilege escalation given unprivileged users administrative rights on the target machine."*

Since most major distributions have already released patches and updates, the best option now would be to install the patches. If that's not immediately feasible for any reason, or if there are no patches available for your distribution, the vulnerability can be prevented from being exploited by removing the SUID bit from the pkexec utility's OS file privileges. And then verify that nothing important was broken by that change. That can be done by using chmod with the '-s' flag to remove the 's' privilege, or by following the NSA's chmod of 0755. (And anyone who doesn't know what any of that means doesn't need to worry about it.)

For anyone who's interested in this sort of hack development, this would be a **perfect** learning opportunity. The Qualys disclosure provides beautiful technical details, deliberately stopping just short of providing a working proof of concept. So, perhaps take what Qualys provided and try your hand at turning that into a working exploit. And for those who want to just see a working exploit for this wonderful (okay, horrifying) vulnerability, here's a link to the .C code:

https://haxx.in/files/blasty-vs-pkexec2.c

And as for this being "re-discovered"?? Believe it or not, Qualys was not the first to discover **exactly** this problem with Polkit's pkexec command. It was originally discovered and blogged about, in full, by an Australian hacker living in Sydney by the name of Ryan Mallon.

https://ryiron.wordpress.com/2013/12/16/argv-silliness/

Ryan's WordPress blog post was titled "argv silliness"

> Most C programmers should be aware that the argv argument to main() is a NULL terminated list of strings, where the first element is the name of the program. On Linux there is an odd "feature" which allows the list to be empty. From the Linux execve(2) manpage:
>
> *On Linux, argv can be specified as NULL, which has the same effect as specifying this argument as a pointer to a list containing a single NULL pointer. Do not take advantage of this misfeature! It is non standard and non portable: on most other UNIX systems doing this will result in an error (EFAULT).*
>
> This allows us to execute an application with argv[0] == NULL. Many applications, including several setuid applications, make the assumption that argv[0] is always a valid pointer. While I haven't found any potential exploits using this, it does allow for some amusing behaviour from setuid binaries.

And after explaining the issue in a bit greater detail, he actually landed on the same exploitable:

> *"After searching around on a stock Ubuntu system for setuid binaries that looked promising for passing argv[0] == NULL to, I found pkexec. pkexec is part of the Polkit package, and allows a binary to be executed as another user (similar to sudo). We call execve()* [Which is a Linux API used to programmatically execute another program] *passing an empty argv list and a single dummy environment variable…"*

So we have another example of Bruce Schneier's sage observation that: "Attacks never get worse, they only ever get better."

Log4J News

Log4Shell hits Ubiquiti

The expected avalanche of Log4j attacks have, so far, failed to materialize. It turns out that the reason for this is that most lower level attackers are looking for out-of-the-box drop-in ready-to-run code. They're not interested in doing a lot of work. And most actually can't. They can run a script and use code that someone else created. But creating it themselves is above their pay grade and what's been seen is that actually exploiting Log4Shell didn't turn out to be that simple. The Log4j library was implemented differently by each app that used it. So there was no universal exploit code that worked everywhere out-of-the-box for everyone, granting attackers the ability to take over systems indiscriminately. Log4j created an opportunity, but it didn't completely provide the means.

But we are seeing instances of Log4j's use in attacks. As we know, attempts have been made against VMWare Horizon, VMWare vCenter, ZyXEL routers, and SolarWinds Serv-U servers. And now the security firm Morphisec has spotted attacks using a customized public exploit for the Log4Shell vulnerability to attack and take over Ubiquiti network appliances running the UniFi software.

The first active exploitation was seen a little over two weeks ago on January 20th using a proof-of-concept exploit which had been previously shared on GitHub. That PoC was developed by Sprocket Security to adapt the Log4Shell exploit to work against Ubiquiti's UniFi devices, and it included complete post-exploitation steps. Don't ask me why Sprocket Security would develop and publicly post such a thing, but that's apparently what they do.

On December 21st they published a blog posting titled: "How to exploit Log4j vulnerabilities in VMWare vCenter" and it begins: "A vulnerability was recently disclosed for the Java logging library, Log4j. The vulnerability is wide-reaching and affects both open source projects and enterprise software, meaning we need to understand how to ID and remediate it in our network environments. Shortly after the issue was disclosed, VMWare announced that several of their products were affected. A Proof of Concept has been released for VMWare vCenter Server instances and explains how this vulnerability allows attackers to execute code as an unauthenticated user using a single HTTP request."

On December 28th they published a blog titled: "Another Log4j on the fire: Unifi" which begins: "By now, you're probably well aware of a recently disclosed vulnerability for the Java logging library, Log4j. The vulnerability is wide-reaching and affects Ubiquiti's Unifi Network Application. In this article, we're going to break down the exploitation process and touch on some post-exploitation methods for leveraging access to the underlying operating system."

And on January 10th they published: "Crossing the Log4j Horizon - A Vulnerability With No Return" which begins: "In this article, we are going to exploit Log4j vulnerabilities in VMWare Horizon, get a reverse shell, and leverage our access to add a backdoor to the VMBlastSG framework. We have also made available a GitHub repository that automates the exploitation process."

Sprocket Security is a penetration testing firm. So I can understand their need to deeply understand the Log4j vulnerability for their own purposes to protect their customers.

But it presents nothing other than harm to the industry for them to publicly post fully working exploit code which is S.K.R. — that's my own new abbreviation for "Script Kiddie Ready."

As we know, Ubiquiti Networks is a very large hardware vendor. Their UniFi software can be installed on Linux and Windows servers to allow network administrators to manage Ubiquiti wireless and networking equipment from a centralized web-based application. In order to be cross-platform, UniFi was built with Java and utilizes the Log4j library for its logging. It was listed as impacted by the Log4Shell and was quickly patched back on December 10th, just one day after the Log4Shell news became public. But we know that patches being available and patches being applied are two different things.

Sprocket Security published its adaptation of the Log4Shell attack for UniFi devices in late December, and Morphisec began seeing attacks starting on January 20th. Morphisec said the attackers took over UniFi devices and ran malicious PowerShell code that later downloaded and installed a version of the Cobalt Strike Beacon backdoor. And researchers noted that this malware communicated with a command and control server that was previously seen attacking SolarWinds Serv-U servers prior to the Log4Shell attacks.

So Log4j and Log4Shell are obviously real. We can be thankful that the exploitation created by Log4j's deliberate URL resolution and dereferencing is tricky and highly application dependent. So no "one-size-attacks-all" exploit is feasible. But helping the bad guys to attack those who have not yet patched — and not waiting even 60 or 90 days, seems the height of irresponsibility.

Zerodium

So much is wrong with this picture. Last Thursday, January 27th, Zerodium added two new entries to what they are calling their "Limited-Time Bug Bounties" for Microsoft's Outlook and Mozilla's Thunderbird. This brings the total of currently active "Temporary Bug Bounties" to three, since Outlook and Thunderbird are joining the longstanding "WordPress PreAuth RCE" which became active on March 31st of last year and remains so today. So until last Thursday, a special bounty for WordPress PreAuth RCE's was all alone.

That WordPress offer explains: *"We are temporarily increasing our payout for WordPress RCEs from $100,000 to $300,000. We are looking for pre-authentication exploits affecting recent versions of WordPress. The exploit should allow remote code execution, work with default installations and should not require any authentication or user interaction."* Of course, the good news for most of the sites using WordPress today, as evidenced by this longstanding and presumably unfulfilled offer, is that the base WordPress is quite secure. It's WordPress' unprofessionally-written add-ons that are the source of all of the havoc. And RCE's involving add-ons do not qualify for this special limited-time offer.

Now, there have been a number of previous "Limited-Time Offer" bug bounties posted by Zerodium. Chrome had an offer for a remote code execution vulnerability which was active from the 14th of September last year through the end of the year. Of that one, Zerodium said: *"We are looking for remote code execution exploits affecting Google Chrome. The exploit should work with Chrome for Android, Windows, Linux and macOS, and support both 32bit and 64bit architectures. Full chains with remote code execution and sandbox escape are eligible for a $1,000,000 bounty."*

Other previous and since expired bounties have been offered for a Chrome Sandbox escape, and against VMware vCenter, Pidgin, ISPConfig, Moodle, IceWarp, SAP NetWeaver and VMware's ESXi. Since a few of these are moderately obscure, as we have in the past, we'd conjecture that some specific client of Zerodium has offered to pay a pretty penny for an exploit against one of those non-mainstream packages. So a special offer was required to focus some "researcher's" attention in that direction.

So now, Zerodium is targeting two of today's most popular and widely used eMail clients with no ending date specified in either case.

For Microsoft's Outlook, Zerodium's offer says: *"We are temporarily increasing our payout for Microsoft Outlook RCEs from $250,000 to $400,000. We are looking for zero-click exploits leading to remote code execution when receiving/downloading emails in Outlook, without requiring any user interaction such as reading the malicious email message or opening an attachment. Exploits relying on opening/reading an email may be acquired for a lower reward."*

"Reward"... yeah.

For Thunderbird, for which, unlike for Outlook they had not been offering any standing bounty, they are now offering $200,000 with the explanation: *"We are looking for zero-click exploits affecting Thunderbird and leading to remote code execution when receiving/downloading emails, without requiring any user interaction such as reading the malicious email message or opening an attachment. Exploits relying on opening/reading an email may be acquired for a lower reward."*

If Zerodium were some beneficent entity working to powerfully incentivize hackers to find the worst of the worst exploits for the purpose of then responsibly disclosing those discoveries to their publishers, I would think that was amazing. But we know what Zerodium is doing. They're a "for profit" Washington D.C.-based enterprise which resells these "rewarded discoveries" to their private state-based clientele. And those discoveries are then used in targeted attacks against others — in direct violation of all cybercrime laws everywhere.

I suppose the creation of Zerodium was inevitable. Wikipedia reports that they pull from a pool of 1,500 researchers and that since its founding in 2015 more than $50 million has been paid out in "reward" bounties.

I'm glad that there are other legitimate channels for reporting and being paid for such discoveries, where those discoveries will be used to repair the affected software rather than to attack unsuspecting and often innocent people, often journalists, dissidents and other "enemies of the state."

"DrawnApart": A device identification technique based on remote GPU fingerprinting
In yet another discovery which can be employed by a web browser's JavaScript, a team of researchers from Australia, France and Israel, including a bunch from the always-industrious Ben-Gurion University of the Negev, have successfully demonstrated yet another brand new fingerprinting technique that exploits a machine's GPU — its graphics processing unit — as a means to track users across the web persistently.

Abstract—*Browser fingerprinting aims to identify users or their devices, through scripts that execute in the users' browser and collect information on software or hardware characteristics. It is used to track users or as an additional means of identification to improve security. Fingerprinting techniques have one significant limitation: they are unable to track individual users for an extended duration. This happens because browser fingerprints evolve over time, and these evolutions ultimately cause a fingerprint to be confused with those from other devices sharing similar hardware and software.*

In this paper, we report on a new technique that can significantly extend the tracking time of fingerprint-based tracking methods. Our technique, which we call DRAWNAPART, is a new GPU fingerprinting technique that identifies a device from the unique properties of its GPU stack. Specifically, we show that variations in speed among the multiple execution units that comprise a GPU can serve as a reliable and robust device signature, which can be collected using unprivileged JavaScript. We investigate the accuracy of DRAWNAPART under two scenarios. In the first scenario, our controlled experiments confirm that the technique is effective in distinguishing devices with similar hardware and software configurations, even when they are considered identical by current state-of-the-art fingerprinting algorithms. In the second scenario, we integrate a one-shot learning version of our technique into a state-of-the-art browser fingerprint tracking algorithm. We verify our technique through a large-scale experiment involving data collected from over 2,500 crowd-sourced devices over a period of several months and show it provides a boost of up to 67% to the median tracking duration, compared to the state-of-the-art method.

DRAWNAPART makes two contributions to the state of the art in browser fingerprinting. On the conceptual front, it is the first work that explores the manufacturing differences between identical GPUs and the first to exploit these differences in a privacy context. On the practical front, it demonstrates a robust technique for distinguishing between machines with identical hardware and software configurations, a technique that delivers practical accuracy gains in a realistic setting.

One of the ways we're succeeding in increasing overall performance is by creating parallel symmetric computation units. This is what the "cores" are in a multiple core system. And in settings where performance and power consumption must be dynamically traded off, we're seeing the development of heterogeneous rather than homogeneous architectures where, for example, a multi-core processor might be composed of a mix of deliberately lower-power and leaner processors which efficiently putt along when not much is happening, and higher-power cores which can be brought to bear only when and as needed.

But in the case of GPUs where there's a team of intended-to-be-identical execution units, one of the coolest things this team did was to exploit today's increasingly parallel approach to set up a deliberate race condition among all available processors, then to monitor the exact sequence of their completion. It turned out that although all of a GPUs execution units are designed to be identical, they are not, and that the exact sequence they finished is stable for any given GPU and different among GPUs. That's the definition of a fingerprint.

They end their paper by discussing responsible disclosure, even though what they found was not a bug. Well, not exactly. They wrote: We shared a preliminary draft of our paper with Intel, ARM, Google, Mozilla and Brave during June-July 2020 and continued sharing our progress with them throughout 2020 and 2021. In response to the disclosure, the Khronos group responsible for the WebGL specification has established a technical study group to discuss the disclosure with

browser vendors and other stakeholders.

Miscellany

Sorting Windows Folders to the TOP!

This is just a random tip that I wanted to share when I finally went Googling and found a solution to something that's been a persistent annoyance for years. When using Windows File Explorer, I have always preferred sorting the listing by date with the most recently modified files at the top. As the command line in UNIX I'll do an "ll -rt" ("rt" for reverse time) which pulls the most recently changed files to the bottom of a list that might scroll off the screen and places them right there near where I am. But in Windows, since the default it usually to show the top of the list, I want the most recently changed files at the top. So I sort by date.

The problem has always been that this places any subdirectories (or "folders" as I suppose we're supposed to call them) at the bottom, which is really annoying when trying to traverse down a hierarchy. And, of course, I could expand the tree hierarchy in the left pane. But what I wanted was the files sorted by date, with the most recent files at the top, AND the folders also at the top.

Anyway, a bit of Googling provided the answer: after clicking on the "Date" header to get the little arrow pointing down, hold the SHIFT key down and click on the "Name" header. PRESTO! I have folders at the top (also sorted from newest to oldest) followed by files sorted from the most to least recent.

And then, of course, I made that the global default by looking under the "Organize" menu and selecting "Folder and Search options" / View / "Apply to Folders" (You might want to do a reset there first to clear out any previous display overrides.)

Anyway... maybe this is dumb and everyone already new this and I was the last one to find out, but I'm so much happier now, and I wanted to share my little discovery with everyone here. :)

Closing the Loop

Igor Lima / @igorlimatweets
Really appreciate your explanation of buffer overflows Steve! These detailed walkthroughs are a main reason I tune in every week. Would love to hear an explanation of how someone could translate such a vulnerability into an RCE. All the best.

Hiveware / @rtischer8277
@GibsonResearch after listening about the NetUSB vulnerability, I checked my own engine code and found the same oversight. Easy to fix now. A nightmare when the code is in the field. Thanks for saving the day with your podcast. Robert Tischer, Falls Church, VA.

Brandt Krueger / @BrandtKrueger
@SGgrc Worried that the authenticator app I've loved and used forever may no longer be being supported by the dev. No updates in years and not responding to support requests. Is there a way to translate codes back into QRs so they can be scanned into a new Auth app?

Itinerant Engineer / @CallMeImperiale
I know how to estimate the entropy in a string of randomly chosen characters (e.g., 20 characters = 128 bits), but how do you estimate the entropy of a string of randomly chosen words, such as "correct horse battery staple"? BitWarden offers passphrases without telling the size of the dictionary. Lance ==)----------

A favorite trick of mine: ln{# of items} / ln2 = equivalent # of binary bits.

SpinRite

In my continuing efforts to give this next release SpinRite every possible useful capability, I've recently been spending some time dealing with mass storage adapters that declare themselves to be RAID controllers so as not to be seen as standard IDE, ATA or AHCI controllers. They provide their own firmware to talk to their hardware. I could have SpinRite ignore the possibility of natively handling the drives attached to those non-standard controllers, just calling upon their firmware to deal with their drives. But SpinRite is able to do a much better job with data recovery and maintenance when it's able to directly access the drive's hardware registers, since that gives SpinRite a much better sense for what's going on. And in almost all cases, except where a controller really **is** offering RAID services, the chipset is actually a 3rd-party chip by ASmedia, JMicron, Marvell, or Silicon Image, all of which present recognizable registers. So now SpinRite is able to work directly with those chips.

SpinRite seems to be about twice as fast in transferring large block of data as AHCI firmware on motherboards or add-on adapters. But in my testing, I've encountered a few instances where SpinRite appears to be a bit slower when talking to bus mastering DMA hardware. The firmware for those adapters may be using some proprietary tricks to get additional speed from them. And since speed is a crucial factor for SpinRite, I need to always use the fastest access method. And that means that SpinRite needs to know which approach will be fastest. So I'm in the final throes of incorporating a mini-benchmark into SpinRite's initial system appraisal, where it locates all of the mass storage attached to a system and works out the best way to talk to each of its drives.

The “Topics” API

After the rather spectacular shunning and failure of Google’s FLoC proposal, their Federated Learning of Cohorts, last week Google unveiled their proposed replacement solution for identifying the real world interests of web browser users for the purpose of presenting more interesting and appropriate ads as a means for increasing click-through rates and thus increasing advertising revenue.

Google has listened and learned and has produced another entirely different and new proposal which attempts to address every reasonable complaint that has been lodged against all previous proposals.

When I say “reasonable complaint” I’m intending to acknowledge that there are some entities, like the EFF, for whom nothing short of absolute and total anonymity will ever be acceptable. The EFF is not only anti-tracking, they are also anti-ad-targeting. They object to any website having any a priori information about its visitors. As such, they stand in opposition to precisely what it is that Google is attempting and hoping to achieve.

So, the as-yet-unanswered question is: Are we going to eventually interact with a world on the web which more resembles the world off the web where everyone driving on the freeway sees the same billboard ads and everyone walking through a shopping mall encounters the same offers? Or, are the unique web technologies, which allow our individual interests to be determined, going to be used to present us with customized content?

That’s a question we’ve addressed before, for which the answer is still to be determined. Since Google wants targeting, and sees that tracking’s days are numbered, they’re focused upon finding a palatable — and that means readily understandable — solution for ad targeting that everyone (other than the EFF) can agree on.

Today we’re going to examine the operation of Google’s next proposal, the Topics API. As we’ll see, if one of FLoC’s terminal weaknesses was that it was largely opaque and incomprehensible, Google apparently learned that lesson well, since “transparency” and “understandabilitty” are Topics’ primary features.

We should start by reminding everyone that Google has an initiative for Chrome which they call “The Privacy Sandbox.” Its headline slogan is: **“Building a more private, open web”** and its headline reads: *“The Privacy Sandbox initiative aims to create web technologies that both protect people’s privacy online and give companies and developers the tools to build thriving digital businesses to keep the web open and accessible to everyone.”*

And, yes, profitable to advertisers. We all know that Google has a vested interest in somehow maintaining advertising relevance. That’s not in question. But they are also running what is, by far, the most popular web browser in the world. Even mighty Microsoft capitulated and adopted Google’s Chromium core. So the fact that they’re interested in using the power of their browser to completely thwart and kill actual tracking — which I think is a far more pernicious threat than ad targeting — is significant. Their Privacy Sandbox initiative has the following three goals:

- **Prevent tracking as you browse the web.**
  People should be able to browse the web without worrying about what personal information is being collected, and by whom. The Privacy Sandbox initiative aims to remove commonly used tracking mechanisms, like third-party cookies, and block covert techniques, such as fingerprinting.

- **Enable publishers to build sustainable sites that respect your privacy.**
  Website developers and businesses should be able to make money from their sites and reach their customers, without relying on intrusive tracking across the web. The Privacy Sandbox initiative is developing innovative, privacy-centric alternatives for key online business needs, including serving relevant ads.

- **Preserve the vitality of the open web.**
  The open web is a valuable resource of information, with a unique ability to both share content with billions of people, and tailor content to individual needs. The Privacy Sandbox proposals aim to both protect your safety online, and maintain free access to information for everyone, so that the web can continue to support economic growth, now and for the future.

Okay, so first a broad overview of the concept behind the new Topics API proposal, then we're going to get down to the details. The Product Director of the Privacy Sandbox for Chrome explains their goals for Topics, as follows:

We started the Privacy Sandbox initiative to improve web privacy for users, while also giving publishers, creators and other developers the tools they need to build thriving businesses, ensuring a safe and healthy web for all. We also know that advertising is critical for many businesses, and is a key way to support access to free content online.

Today, we're announcing Topics, a new Privacy Sandbox proposal for interest-based advertising. Topics was informed by our learning and widespread community feedback from our earlier FLoC trials, and replaces our FLoC proposal.

With Topics, your browser determines a handful of topics, like "Fitness" or "Travel & Transportation," that represent your top interests for that week based on your browsing history. Topics are kept for only three weeks and old topics are deleted. Topics are selected entirely on your device without involving any external servers, including Google servers. When you visit a participating site, Topics picks just three topics, one topic from each of the past three weeks, to share with the site and its advertising partners. Topics enables browsers to give you meaningful transparency and control over this data, and in Chrome, we're building user controls that let you see the topics, remove any you don't like or disable the feature completely.

More importantly, topics are thoughtfully curated to exclude sensitive categories, such as gender or race. Because Topics is powered by the browser, it provides you with a more recognizable way to see and control how your data is shared, compared to tracking mechanisms like third-party cookies. And, by providing websites with your topics of interest, online businesses have an option that doesn't involve covert tracking techniques, like browser fingerprinting, in order to continue serving relevant ads.

The overall concept is simple: The browser notices the websites being visited and looks up a set of topics from a curated list of available topics. That list currently contains 350 topics:

https://github.com/jkarlin/topics/blob/main/taxonomy_v1.md (https://grc.sc/856)

The browser calculates the top topics for its user based upon their recent browsing activity and it provides a JavaScript API which provides topics currently of interest to the user, to help enable the selection of appropriate ads. But there's also a lot more to it than that as we're about to see.

The intent of the Topics API is to provide the users of the API, typically 3rd-party ad-tech or advertising providers on the page running the JavaScript, with coarse-grained (and as we'll see, deliberately fuzzed and privacy-enhanced) advertising topics that the page visitor might currently be interested in. These topics will supplement the contextual signals from the current page (is it for planning weddings or vacations).

Google calls this list their advertising taxonomy. It will consist of somewhere between a few hundred to a few thousand topics and it will deliberately exclude sensitive topics such as gender and race. Google says that they're planning to engage with external partners to help define and refine the list. Their eventual goal is for the taxonomy to be sourced by an external party that incorporates feedback and ideas from throughout the industry.

The topics will be inferred by the browser by using a classifier to map site domain names to topics. The classifier's weighting will be public, perhaps built by an external partner, and will improve over time. In some brainstorming, Google has suggested that it might make sense for sites to provide their own topics via meta tags, headers, or JavaScript... but that remains an open question for later discussion.

For any given web domain the classifier may return no topics, or it may return one or several. There is no limit, though the expectation is 1-3. So it's possible that a site may map to no topics and so doesn't add to the user's topic history. Or it's possible that a site adds several topics or increases the popularity of the user's existing topics.

In the past, we've talked about the DOM, the standardized Document Object Model. One of the objects that's always present in this model is the "document" object. The document object has properties like "body" which is the document's body text, "images" which is a collection of all of the images present in the document, and "links" which is a collection of all of the links present in the document. The Topics API adds the "browsingTopics" property to every page's document object.

"browsingTopics" provides anyone who asks with up to three topics from the ~350 item taxonomy, one from each of the preceding three weeks, and those three topics are returned in random order. No effects from a user's web activity persists past the 3rd week. When a visitor visits a site, it will be able to obtain up to three topics which might help it or its advertisers to choose a more relevant ad. It was decided to provide three topics so that a site that doesn't see visitors often will still obtain sufficient information about the user to choose something useful. And a granularity of one week between updates was chosen so that sites (and advertisers) which are seen more often will learn at most one new topic per week.

And what's more, not all sites see the same weekly topic from any given user. Here's how that works:

For each week, the user's top 5 topics, obtained from the domains they visited during the preceding week, are determined from browsing information local to the browser — at no time does the user's browser contact any external servers for help with choosing. So, five master topics are chosen from an examination of the preceding week's browsing history. And one additional topic is chosen completely at random from the taxonomy.

Then, when the "document.browsingTopics()" API is called by a site the user is visiting, or JavaScript running in one of its ads, the topic returned for each of the three weeks (which will be returned in random order) is chosen from the 6 available topics as follows:

With a 5% chance (1 in 20) the randomly chosen topic will be returned.

Otherwise the value of an HMAC hash is computed from a static per_user_private_key, the week number, and the document page's website domain name. That HMAC hash is then taken modulo 5 to produce a uniformly distributed value from 0 to 4, which is used to choose one of the user's five top sites for that week.

This may seem overly complicated, but it's clever and privacy enforcing: The use of an HMAC keyed by a per-user secret, the week number, and the domain of the webpage, means that for that week, the user's browser will always present the same one-of-five Topics to anyone querying at the same site, but that every site will see an unpredictable but constant one-of-five topics for that user for that week.

This minimizes the amount of information being disclosed, since no site will receive more than one fixed topic per user per week. And each site gets only one of the five real topics to make it much more difficult to cross-correlate the same user.

Google introduces the 5% noise to ensure that each topic has a minimum number of members as well as to provide some amount of plausible deniability, meaning that no topic can be regarded as absolute — it may have been deliberately chosen at random.

And finally, the point in time where a user's week ends and the next one begins is fixed but also chosen at random. So this introduces some additional uncertainty and noise since not everyone's web browser will calculate new topics at the same time on the same day.

One last VERY tricky and important bit...

Now, there's one last extremely tricky bit that's a bit of a mind bender. But it's an important privacy safeguard for the entire system. It addresses the problem that FLoC had and which Leo mentioned several times on other TWiT podcasts. The problem was that FLoC was broadcasting a token which was a condensation of the person's web browsing history, and thus by extension a condensation of them. And those who knew how to interpret the token would know what it meant. And this token was being presented to any website they visited without prompting. This was correctly seen as delivering a significant **reduction** in user privacy.

And the problem did not fall upon deaf ears. Google has clearly given this a great deal of thought and the Topics API incorporates a mitigation for this problem. But first I should mention that there's nothing whatsoever salacious or even really very interesting about the list of topics. They are borrrrrrring!! But they do make sense from an advertiser's standpoint. So the first point is there's just no way for anything very personal to be revealed or represented by these topics. It's this week's GRC shortcut to make it easy to find: https://grc.sc/856. Have a look for yourself!

But even so, the "Topics" system contains a strong topics filter. Here's how it works:

Not every caller of this API will receive all of a user's three chosen browser topics. Only API callers that observed the user's browser visiting some site which mapped to the topic in question within the prior three weeks qualifies to receive the topic. In other words, if the user of the API — website or advertiser on a site — did not call the API sometime in the past three weeks for that user's browser when they were visiting a site which mapped to the topic in question, then the topic will be filtered and not included in three-topic the array returned by the API.

This is a bit mind bending, so let's use an example.

During the previous couple of weeks a user has been browsing a lot about travel. So for the time being the browser has learned that about them and chosen to represent their travel-related interest to the world as they visit other sites. So now suppose that they're at a site about gaming and an advertiser on that site runs JavaScript in an ad insertion frame which queries the document.browsingTopics API to receive the three topics of interest to the user—in no particular order—which the user's browser has chosen to offer anyone querying about its user while on this gaming site… (Remember that each week the top five interest topics are chosen and one of those five is selected from each of the previous three weeks based upon a hash of the domain name being visited.)

ONLY IF THAT advertiser had queried THAT user's document.browsingTopics API sometime within the previous three week's WHILE that browser was at some site whose topics matched the topic the user's browser had chosen to offer at this site now, would that topic not be filtered and thus would be presented to that advertiser.

In other words, in order to obtain an interest topic from a user when they are wandering around anywhere on the Internet, an advertiser must have previously asked their browser about them during the past three weeks while they were on a site whose topics may have contributed to the topic they are offering this week.

In this world, assuming that we've blocked 3rd-party cookies, fingerprinting and all other tracking — and perhaps even outlawed it, which admittedly may be what it takes — the advertiser doesn't know who the user is. But they will have recently pinged the user's browser while the user was at a website whose topics matched the topic the user is now offering.

Google explains that this extra topic information filtering is intended to "prevent the direct dissemination of user information to more parties than the technology that the API is replacing" — in other words, third-party cookies. Another way of phrasing it is that this ensures that 3rd-parties don't learn more about a user's past than they could with cookies.

And when you think about it, it does that. As I started out saying, it prevents the problem that FLoC presented of simply blabbing about a user's previous website history via a weird hashed token.

The history window which limits the inclusion of topics available to each caller is three weeks.

It should be obvious, but only topics of sites that use the API, or host ads that query the API, will contribute to the weekly calculation of topics.

Also, only sites that were navigated to via user gesture are included, as opposed to a redirect, for example. So it's not possible to bounce users around to load-up their browser with topics.

If the API cannot be used for any reason, if it's disabled by the user or by a response header, then the page visit will not contribute to the weekly calculation.

If the user opts out of the Topics API by disabling it in browser settings, or is in incognito mode, or the user has cleared their browser history, no topics will be returned.

So, the goal of the Topics API's is to take a step forward toward increased user privacy, while still providing enough relevant information to advertisers that websites can continue to thrive, but without the need for invasive tracking enabled via existing tracking methods.

Another important aspect of the Topicss API is the user's understanding and sense of control.

Google understands that their users should be able to understand the purpose of the Topics API, recognize what information is being provided about them, know when the API is in use, and be provided with controls to enable, disable, or edit it.

The API's human-readable taxonomy enables people to learn about and control the topics that may be suggested about them by their browser. Users can remove topics they specifically do not want the Topics API to share with advertisers or publishers, and there can be user experience for for informing the user about the API and how to enable or disable it. Chrome would provide information and settings for the Topics API at chrome://settings/privacySandbox. In addition, topics are not available to API callers in Incognito mode, and topics are cleared when browsing history is cleared.

Moreover, unlike FLoC, which built its hash from every site visited, only sites that include code which calls the Topics API would be included in the browsing history eligible for topic frequency calculations. In other words, sites are not eligible for contributing to topic frequency calculations unless the site or an embedded service has taken the action of calling the Topics API.

Site will be allowed to block topic calculation for their visitors using Permissions-Policy header:

Permissions-Policy: browsing-topics=()

And, no topics will ever be returned if:

- The user opts out of the Topics API via browser settings at chrome://settings/privacySandbox.

- The user has cleared their topics (via the browser settings at chrome://settings/privacySandbox) or cleared their cookies.

- The browser is in Incognito mode.

So, that's the operation of Google's newly proposed Topics API. It feels like a FAR more refined effort than FLoC. Its implementation will require VASTLY more work at the browser end than anything before, but that feels appropriate, too, if Google wants to hold onto advertiser buy-in while working to eliminate 3rd-party tracking.

We know what the EFF would say. They want nothing less than pure and absolute anonymity. I have no way to gauge how much revenue websites would lose if targeted advertising were eliminated. But we hear that it would be a significant blow. I certainly wouldn't shed a tear over the end of any companies whose entire business model is secretly tracking users against their wishes.

The Topics API feels as if Google is finally getting really serious about offering a compromise to pure tracking which advertisers can live with — they're a huge one, after all — and which offers sufficient clarity, visibility, feedback and control that users should feel comfortable, too.

I like it.

If it's just added to the tracking and fingerprinting mess that we already have, then we lose. But if its adoption allows Google to join Firefox in truly battling 3rd-party cookies with per-site siloing and other proactive anti-tracking and anti-fingerprinting measures... that is, if Google really an finally have their heart in it, then I think it has to potential to be a big step forward for the industry.