

Security Now! #1056 - 12-16-25

Australia

This week on Security Now!

- Home Depot's puzzling reluctance to close a bad hole.
- GNOME's shell extension manager is unhappy with AI.
- How attacks on open source repositories compares in 2025.
- China's researchers have taken aim at the US power grid.
- How bad has the React2Shell vulnerability turned out to be.
- More new React vulnerabilities.
- Apple moves to iOS 26.2.
- Let's Encrypt's crosses into one billion servers managed.
- A DNS Benchmark update.
- Some interesting listener feedback, then...
- How things going with Australia's social media ban and what we are learning.



Security News

Home Depot reluctantly removes exposed access

Last week I used the phrase “Oh yeah? Well make me!” in reference to the sort of conduct that’s probably most common in adolescent males of around high school age. I was reminded of that by TechCrunch’s reporting of Home Depot apparently taking the same unfortunate tactic. TechCrunch’s headline was “Home Depot exposed access to internal systems for a year, says researcher”. Zack Whittaker reported:

*A security researcher said Home Depot exposed access to its internal systems for a year after one of its employees published a private access token online, likely by mistake. The researcher found the exposed token and tried to privately alert Home Depot to its security lapse but was ignored for several weeks. The exposure is now fixed **after** TechCrunch contacted company representatives last week.*

Security researcher Ben Zimmermann told TechCrunch that, in early November, he found a published GitHub access token belonging to a Home Depot employee, which was exposed sometime in early 2024. When he tested the token, Zimmermann said that it granted access to hundreds of private Home Depot source code repositories hosted on GitHub and allowed its holder to modify their contents.

Yikes! Just to pause here for a second, we don’t know what those “hundreds of private Home Depot source code repositories” might have contained. But having a token loose on the Internet that permits write-access to them ought to keep anyone from resting before it was invalidated.

We haven’t encountered this Ben Zimmermann before, but Zack provided a link to Ben’s website where he introduces himself, writing: “Hey, I’m Ben. I’m a security researcher from California. I’ve been awarded over \$20,000 in bug bounties for securing critical infrastructure and the open web.” And he lists a bunch of his discoveries. So he appears to be the real deal. Zach continues:

The researcher said the keys allowed access to Home Depot’s cloud infrastructure, including its order fulfillment and inventory management systems, and code development pipelines, among other systems. Home Depot has hosted much of its developer and engineering infrastructure on GitHub since 2015, according to a customer profile on GitHub’s website.

Zimmermann said he sent several emails to Home Depot but never heard back. Nor did he get a response from Home Depot’s chief information security officer, Chris Lanzilotta, after sending a message over LinkedIn. Zimmermann told TechCrunch that he has disclosed several similar exposures in recent months to companies, which have thanked him for his findings. He said: “Home Depot is the only company that ignored me.”

Given that Home Depot offers no way to report security flaws, such as a vulnerability disclosure or bug bounty program, Zimmermann contacted TechCrunch in an effort to get the exposure fixed. When reached by TechCrunch on December 5, Home Depot spokesperson George Lane acknowledged receipt of our email but did not respond to follow-up emails asking for comment. The exposed token is no longer online, and the researcher said the token’s access was revoked soon after our outreach.

We also asked Lane if Home Depot has the technical means, such as logs, to determine whether anyone else used the token during the months it was left online to access any of Home Depot’s internal systems. We did not hear back.

That question at the end of Zack's reporting is exactly the one I was asking myself. Ben was able to date the creation of the token to early 2024. So we're coming up on two years of write-access exposure to many of Home Depot's critical-appearing internal systems by way of the software that runs them. Ben is a good-guy security researcher who is out there working to improve the security of the world. But we know that within the population of people who may be poking around looking for security vulnerabilities, good guys like Ben are almost certainly in the minority. Access to hundreds of Home Depot's internal operations source code repositories would be immensely valuable to any attacker who wants to find some way to threaten and extort Home Depot, a well known US entity with deep pockets and apparently not much in the way of security practices. So do they have logs? Do they even care if they have them? We don't know anything about Home Depot's internal IT culture, but what we do know doesn't look good.

GNOME's not happy about AI generated Shell Extensions

A recent posting by one of the guys who has taken on the job of managing GNOME's shell extensions was interesting and I wanted to share it. First of all, just to be clear about what GNOME is for those who may be primarily familiar with the Windows or Mac world, GNOME is to Linux and Unix-like operating systems what Explorer and the Windows desktop is to Windows, or Finder and the macOS UI is to macOS. All three are the primary graphical environment users interact with daily. GNOME — G. N. O. M. E. — was originally an abbreviation for GNU Network Object Model Environment. Since then it's taken on a life of its own and it's just GNOME. Therefore, a GNOME shell extension is an add-on that adds a feature to the Linux desktop which runs GNOME. So here's what one of the Shell Extension managers wrote last week:

Since I joined the extensions team, I've only had one goal in mind. Making the extension developers' job easier by providing them documentation and help. I started with the port guide and then I became involved in the reviews by providing developers code samples, mentioning best practices, even fixing the issue myself and sending them merge requests. Andy Holmes and I spent a lot of time writing all the necessary documentation for the extension developers. We even made the review guidelines very strict and easy to understand with code samples.

Today, extension developers have all the documentation to start with extensions, a port guide to port their extensions, and a very friendly place on the GNOME Extensions Matrix channel to ask questions and get fast answers. We now have a very strong community for GNOME Shell extensions that can easily overcome all the difficulties of learning and changes.

The number of submitted packages is growing every month and we see more and more people joining the extensions community to create their own extensions. Some days, I spend more than 6 hours a day reviewing over 15,000 lines of extension code and answering the community.

In the past two months, we have received many new extensions. This is a good thing since it can make the extensions community grow even more, but there is one issue with some packages. Some devs are using AI without understanding the code being produced. This has led to receiving packages with many unnecessary lines and bad practices. And once a bad practice is introduced in one package, it can create a domino effect, appearing on other extensions. That alone has increased the waiting time for all packages to be reviewed.

At the start, I was really curious about the increase in unnecessary try-catch block usage in many new extensions being submitted. So I asked, and they answered that it is coming from AI. Just to give you a gist of how this unnecessary code might look:

```

destroy() {
  try {
    if ( typeof super.destroy === 'function' ) {
      super.destroy();
    }
  } catch (e) {
    console.warn(` ${e.message} `);
  }
}

```

Instead of simply calling `super.destroy()`, which you clearly know exists in the parent:

```

destroy() {
  super.destroy();
}

```

At this point, we have to add a new rule to the review guidelines: Any packages with unnecessary code that indicate they are AI-generated will be rejected.

This doesn't mean you cannot use AI for learning or fixing some issues. AI is a fantastic tool for learning and helping find and fix issues. Use it for that, not for generating the entire extension. For sure, in the future, AI can generate very high quality code without any unnecessary lines. But until then, if you want to start writing extensions, you can always ask us in the GNOME Extensions Matrix channel.

In order to understand this manager's annoyance and also so that we can then talk about what AI is doing here, we need to dig a bit into some interesting coding weeds.

Modern high level languages have a construction known as Try/Catch. (Leo will be glad to know that this concept originated in the 1960's with LISP, which used the semantics "Catch" and "Throw".) The idea is that if some code might produce an error at run time, we don't want the entire program to just give up and explode. We want to have the opportunity to contain the problem and to possibly handle it ourselves. So, the suspect code is placed inside a "Try" block which tells the runtime manager to literally "Try" this. The "try" block is followed by a "catch" block that's used to "catch" any runtime error that might unexpectedly occur while we're executing code inside the "try" block. In other words, we're telling the runtime manager: While code is executing inside this "try" block, don't freak out if anything bad happens. Simply stop what you're doing and execute the code we've provided for this purpose in the "catch" block which immediately follows the "try" block, and we'll take it from there. This allows code to be somewhat self healing and to handle its own errors internally rather than simply crashing.

Okay. So now let's look at the specific case of this gratuitous AI-generated code: In the example this manager provided, we have some code in the "try" block that, first of all, cannot possibly fail; it's already being extremely cautious. It first checks to see whether the "super" object contains a function named "destroy()". The test for that, just asking the question "does this exist?" cannot produce an error. It will either return true or false – either the "super" object exposes a "destroy()" function or it doesn't. And then, the way the conditional is written, only if the "super" object exposes a "destroy()" function will that "destroy()" function then be called on the "super" object. So this conditional that's wrapped in a "try" block cannot possibly fail. But more than this, we learn from the context that the manager says whatever the

"super.destroy()" function is, it is apparently well known to exist and must exist in this GNOME Shell extensions environment. That makes it always safe to simply call the function. It will always be present and simply calling it cannot ever fail.

So not only was the use of that Try/Catch construction provably and obviously unnecessary, because the conditional expression it contained was first testing for the presence of the function and only calling the function if it existed. But that conditional test itself was also completely superfluous because whatever that super.destroy() function might be, apparently it must always be present. That means that everything there, all of that code, other than simply calling the super.destroy() function directly was superfluous gratuitous nonsense.

So how did this happen? It happened because today's LLM-based AI doesn't understand, even a little bit, what it's doing. It doesn't know whether we're asking about the population of kangaroos in Australia or asking for code to destroy the super object. It's all the same to today's AI. It's all just language. Which brings us to the main point of this:

The thing that I thought was most interesting was the observation that AI-generated code could and would become "infected" with nonsense code, like this. That's a very interesting observation that, I'm sure, tracks with everyone's intuitive and growing understanding of the way today's LLM based AI operates. Today's AI is all just astonishingly sophisticated pattern matching. So somewhere along the way, AI picked up that conditional test construction of making sure that a function existed on an object before calling that function. It doesn't hurt to do that, but our code would get seriously bogged down if we were to keep asking the runtime manager to verify the presence of known existing functions before every time we call one. The point is that testing like this for a function that might **not** exist is a good thing to do – so there is a place for it and AI picked up on that useful instance – without any understanding of why, and is now salting the code it produces with that nonsense without need. Alternatively, you could protect yourself from a missing function at runtime by wrapping the function call in a Try/Catch construction. We saw that too. So in this case the AI did both when neither were necessary.

So here's where the notion of infection comes in, which is from promulgation: We know that AI is training on what it finds out on the open Internet, even if what it finds is code that it or some other AI previously emitted. That means that superfluous code like what we've just seen, which doesn't cause errors but adds nothing other than overhead and bloat, will tend to be self-perpetuating. If this manager didn't proactively strip this crap out of GNOME's open source shell extensions code base, it would remain there to be picked up by LLMs that would further replicate it into the future. And the more it's replicated the stronger it becomes. The pattern takes hold. Before long, code would be littered with this, because non-coders would be asking AI to write an extension without ever bothering, or needing, to look inside.

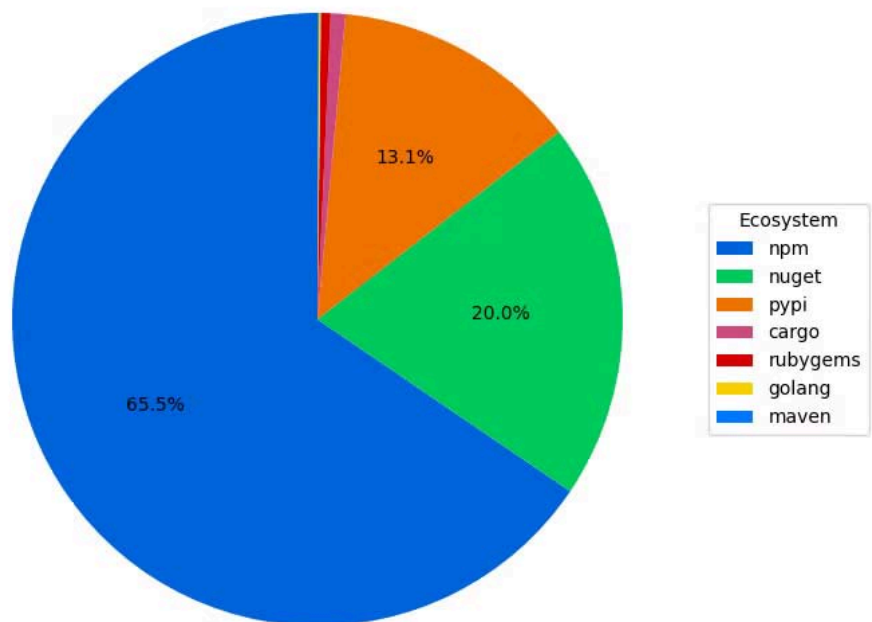
I wanted to spend some time on this because it clearly represents a danger to open source code. The crucial thing to appreciate is that AI is producing code without understanding it. And any code that doesn't cause an error that forces its correction will not be corrected or removed.

That said, all is not lost forever, because I am 100% absolutely certain that future coding AI, that's been specifically designed for coding, will look at the code that was emitted by these early LLMs and just shake its head. It would be able to see and actually understand the code used in this example. It would know that the super.destroy() function must always be present in the super object. So it would remove the conditional test for its presence. Then it would see that what's left in the "Try" block cannot possibly fail, so it would completely remove the Try/Catch construction and all of the code from the "catch" block which could never be executed.

2025 Repository Malware – Year in review:

The deliberate pollution of our industry's open source repositories by malicious actors is one of the most unfortunate but retrospectively obvious problems of the open source movement. The altruistic goal is to allow all well-meaning coders to share and share alike. It's a wonderful concept in principle. But nothing has ever been more prone to abuse because the entire system is built on the assumption of good will by those who are contributing.

As the year 2025 draws to a close, we're able to look back on this year to compare it to 2024.



As usual, the volume of packages submitted to NPM – the package management repository for JavaScript – in 2025, far outweighs what's seen in any of the other repository ecosystems. The primary reason for this is that when looking at web applications, regardless of the backed technology (e.g. Java, Rust, C#, etc.), it is most common for the front-end UI to be built using JavaScript or TypeScript. These front-end technologies largely depend upon NPM. Adding to this is the fact that it's very straightforward to author and publish packages to NPM, which explains why there's a consistently high level of activity there.

As shown by the pie chart I included in the show note, NPM holds 2/3rds of the entire repository segment. NuGet holds 2nd place being the repository for the .NET ecosystem with 20% share to NPM's 65.5%. Third place is PyPi with 13.1%, followed by, in order of decreasing share, cargo, rubygems, go, and maven. Taken together, those top three of NPM, NuGet and PyPi comprise 98.5% of the entire space with those also-rans holding only 1.5%.

Overall, there was an 86.8% increase in malicious submissions of all kinds relative to the same period last year. To give everyone some sense for this, here are the counts and natures of what bad guys were hoping to slip into the repositories and slip into other users' and developers' code:

- **4,196** packages were specifically designed to target groups or organizations, often linked to cyber espionage or financial theft.
- **58,473** packages contained URLs known to be malicious, underscoring the growing risk of dependency injection attacks.
- **929,789** packages included pre-compiled binaries, creating potential attack vectors for binary tampering.
- **160,959** packages executed suspicious code during installation.
- **38,092** packages made server requests to IP addresses, attempting to communicate with command-and-control servers.

- **1,062,697** packages attempted to obfuscate their underlying code, making detecting malicious activity much more difficult.
- **4,863** packages were identified as typosquats, indicating a concerted effort by attackers to trick developers into installing malicious versions of popular packages.
- **+61,801** spam packages were published across ecosystems, severely degrading the integrity of open-source repositories and threatening the trust developers place in these platforms.
- **206,632** packages were flagged as containing critical malware, requiring immediate attention.

The VersCode group, who make it their business to keep an eye on all this, wrote about the trends they have seen changing over the past year. They said:

We observed several trends across these categories of malicious behavior when compared to last year. Most notably, it is now common for packages to make use of obfuscation, a normally benign technique used to make the code harder to analyze or reverse engineer in order to protect Intellectual Property (IP). However attackers are leveraging this to disguise malicious payloads and make detection significantly more difficult.

We saw a rise in code that executes during package installation. This is particularly problematic for malware analysis when malicious code is fetched from outside the package itself, for example, via a file downloaded from a URL during installation using pre/post-install hooks. This dynamic nature makes it hard to be certain whether a package is malicious or not as the contents of the file behind the URL could change at any time, swapping out a benign or legitimate file for a malicious payload.

There was a reduction in dependency confusion attacks this year, suggesting tactics to target specific groups or organizations for financial gain have changed, and other more effective means are being used instead.

There's no easy solution to this. The repositories are so popular because they serve as a source of terrific ready-made code that solves problems for developers. The only thing developers can do is remain vigilant and inspect anything that's downloaded. Unfortunately, because benign behavior can change, even an initial all-clear might not be enough caution.

China is really taking this seriously

We've recently talked about various countries becoming worried after their discovery of multiple undocumented cellular radios hidden inside their widely deployed Chinese-made electric buses. In the first case that we reported, the buses were driven into what were described as bus-sized Faraday cages to cut them off from any outside monitoring or control and all of the SIM cards were removed from their secret cellular radios.

The news of these buses wouldn't have surprised any of our long-term listeners since we had previously reported upon the similar discovery of undocumented cellular radios in Chinese made dock-side shipping cranes and the inverters used to convert the DC current produced by wind turbine and solar panels into AC.

So, we have all of that; and it's a lot. And we might think that it would be difficult to further surprise and worry us. But then you learn that from 2010 to the present, Chinese researchers have published 2,723 research papers – most never translate into English – on the subject of vulnerabilities in the United States power grid. 2,723 with at least 225 of those papers, not quite 10%, but still 225, which explicitly explore potential attacks on the US power grid. I sincerely hope that people over on this side of the Pacific who are in a position to do something about this are also studying these papers and not sitting around waiting for something bad to happen.

Strider Intelligence's report is titled: "*In Broad Daylight: U.S. Grid Exposed to Risk from PRC-Manufactured Inverter Equipment*" They wrote:

The People's Republic of China (PRC) is systematically targeting America's critical infrastructure as part of a long-term strategy to gain leverage in a crisis. These are coordinated campaigns to pre-position access across the systems that keep the U.S. running.

This new report from Strider details the United States' growing dependence upon inverter-based resources, including solar inverters and battery energy storage systems, manufactured by companies in the People's Republic of China. These networked, software-driven devices are capable of remote communication and control which, when combined with their PRC origin, expose U.S. critical infrastructure to unprecedented risk.

Under the PRC's 2017 National Intelligence Law, any domestic company can be compelled to support state intelligence activities. As a result, PRC-made inverter-based resources inherently carry elevated security risks, regardless of direct ties to high-risk entities.

Strider's analysis found that nearly half of all inverters and battery energy storage systems imported into the United States between 2015 and 2024 came from a high-risk PRC manufacturer. Additionally, 86% of U.S. utilities surveyed for this report (representing about 12% of installed U.S. capacity) rely on at least one risky PRC supplier in their power composition. Three of the high-risk PRC suppliers found were:

- *Contemporary Amperex Technology (CATL): In 2025, the U.S. Department of Defense labeled CATL a "Chinese military company," flagging national-security and sanctions exposure.*
- *Huawei: The company has a documented history of IP theft accusations, export control violations, and close alignment with the PRC military, intelligence and law enforcement entities. Huawei was added to the U.S. Commerce Department's Entity List and banned from U.S. 5G networks due to espionage risks, but there is no federal rule banning Huawei solar inverters.*
- *Sungrow: The company's CEO and Chairman is a member of the National People's Congress (the legislative body of the PRC state) and nearly 30% of Sungrow's senior management are Chinese Communist Party (CCP) members.*

Within the 2,723 PRC research publications examining weaknesses in the U.S. energy grid, at least 225 of those publications related to potential attacks against the U.S. grid—including multiple publications that ran attack simulations on the western U.S. power grid to test new concepts, methods, and tools.

China and the U.S. have the most bizarre interdependent relationship. Perhaps co-dependent would be a better term. I don't understand. But then I also don't understand the world's super

powers have their nuclear arsenals aimed at each other. That's crazy too. So perhaps this cyber-war nonsense is much the same as the nuclear standoff that has been in place for decades. Let's just hope that no one ever makes the mistake of pulling any triggers.

The state of the React2Shell vulnerability

Following up on last week's podcast, which I titled "React's Perfect 10", last Friday the 12th, Google updated the world on the five Chinese state actors that have been seen actively attacking the West through this distressingly easy-to-exploit and widespread vulnerability in REACT servers. Google's Friday posting was titled: "*Multiple Threat Actors Exploit React2Shell (CVE-2025-55182)*" and they wrote:

On Dec. 3, 2025, a critical unauthenticated remote code execution (RCE) vulnerability in React Server Components, tracked as CVE-2025-55182 (aka "React2Shell"), was publicly disclosed. Shortly after disclosure, Google Threat Intelligence Group (GTIG) had begun observing widespread exploitation across many threat clusters, ranging from opportunistic cyber crime actors to suspected espionage groups.

GTIG has identified distinct campaigns leveraging this vulnerability to deploy a MINOCAT tunneler, SNOWLIGHT downloader, HISONIC backdoor, and COMPOOD backdoor, as well as XMRIG cryptocurrency miners, some of which overlaps with activity previously reported by Huntress. These observed campaigns highlight the risk posed to organizations using unpatched versions of React and Next.js. This post details the observed exploitation chains and post-compromise behaviors and provides intelligence to assist defenders in identifying and remediating this threat.

Google then reminds about the nature and background of the React problem which I'll skip since we covered that at length last week. What I think is interesting and important is to look at what Google is actually seeing being done, enabled by this perfect 10 vulnerability. It's one thing to say "Oh, that's not good!" but it's still useful to see exactly what that means. They write:

Since exploitation began, GTIG (Google Threat Intelligence Group) has observed diverse payloads and post-exploitation behaviors across multiple regions and industries. In this blog post we focus on China-nexus espionage and financially motivated activity, but we have additionally observed IranIan based actors exploiting CVE-2025-55182.

As of Dec. 12, GTIG has identified multiple China-nexus threat clusters utilizing CVE-2025-55182 to compromise victim networks globally. Amazon Web Services reporting indicates that China-nexus threat groups Earth Lamia and Jackpot Panda are also exploiting this vulnerability. GTIG tracks Earth Lamia as UNC5454. Currently, there are no public indicators available to assess a group relationship for Jackpot Panda.

Okay. So now let's look at the actual exploitations:

MINOCAT — *GTIG observed China-nexus espionage cluster UNC6600 exploiting the vulnerability to deliver the MINOCAT tunneler. The threat actor retrieved and executed a bash script used to create a hidden directory (\$HOME/.systemd-utils), kill any processes named "ntpclient", download a MINOCAT binary, and establish persistence by creating a new cron job and a systemd service and by inserting malicious commands into the current user's shell config to execute MINOCAT whenever a new shell is started. MINOCAT is a 64-bit ELF executable for*

Linux that includes a custom "NSS" wrapper and an embedded, open-source Fast Reverse Proxy (FRP) client that handles the actual tunneling.

It helps, I think, to appreciate that these are not theoretical attacks. They are actually happening to people. If this happens to a server, the Fast Reverse Proxy client phones home to establish a persistent connection, allowing bad actors – apparently Chinese bad actors – to do whatever they wish with the compromised system. And the important thing to appreciate is that this is all a persistence mechanism. The owner of the React server might have then patched it. But it was too late. The machine has already been owned.

SNOWLIGHT — *In separate incidents, suspected China-nexus threat actor UNC6586 exploited the vulnerability to execute a command using cURL or wget to retrieve a script that then downloaded and executed a SNOWLIGHT downloader payload. SNOWLIGHT is a component of VSHELL, a publicly available multi-platform backdoor written in Go, which has been used by threat actors of varying motivations. GTIG observed SNOWLIGHT making HTTP GET requests to C2 (Command & Control) infrastructure to retrieve additional payloads masquerading as legitimate files.*

GTIG also observed multiple incidents in which a different threat actor UNC6588 exploited the vulnerability, then ran a script that used wget to download a COMPOOD backdoor payload. The script then executed the COMPOOD sample, which masqueraded as Vim. GTIG did not observe any significant follow-on activity, and this threat actor's motivations are currently unknown.

COMPOOD has historically been linked to suspected China-nexus espionage activity. In 2022, GTIG observed COMPOOD in incidents involving a suspected China-nexus espionage actor, and we also observed samples uploaded to VirusTotal from Taiwan, Vietnam, and China.

HISONIC — *Another China-nexus actor, UNC6603, deployed an updated version of the HISONIC backdoor. HISONIC is a Go-based implant that utilizes legitimate cloud services, such as Cloudflare Pages and GitLab, to retrieve its encrypted configuration. This technique allows the actor to blend malicious traffic with legitimate network activity. In this instance, the actor embedded an XOR-encoded configuration for the HISONIC backdoor delimited between two markers, "115e1fc47977812" to denote the start of the configuration and "725166234cf88gxx" to mark the end. Telemetry indicates this actor is targeting cloud infrastructure, specifically AWS and Alibaba Cloud instances, within the Asia Pacific (APAC) region.*

*Finally, we also observed a China-nexus actor, UNC6595, exploiting the vulnerability to deploy **ANGRYREBEL.LINUX**. The threat actor uses an installation script (b.sh) that attempts to evade detection by masquerading as the legitimate OpenSSH daemon (sshd) within the /etc/ directory, rather than its standard location. The actor also employs timestomping to alter file timestamps and executes anti-forensics commands, such as clearing the shell history (history -c). Telemetry indicates this threat actor cluster is primarily targeting infrastructure hosted on international Virtual Private Servers (VPS).*

So I think it's important to appreciate that real people and organizations are being hurt due to this vulnerability. What's unclear to me, as it was last week when we first reported on this, is why the updated React server code update was not given a great deal more time to filter out into the React server installed base before it was disclosed publicly to trigger the feeding frenzy. Perhaps any update to React would have triggered an investigation and reverse engineering of the changes by malign forces, so that it was better to make a big noise. But this was a bad one.

More (new) React vulnerabilities

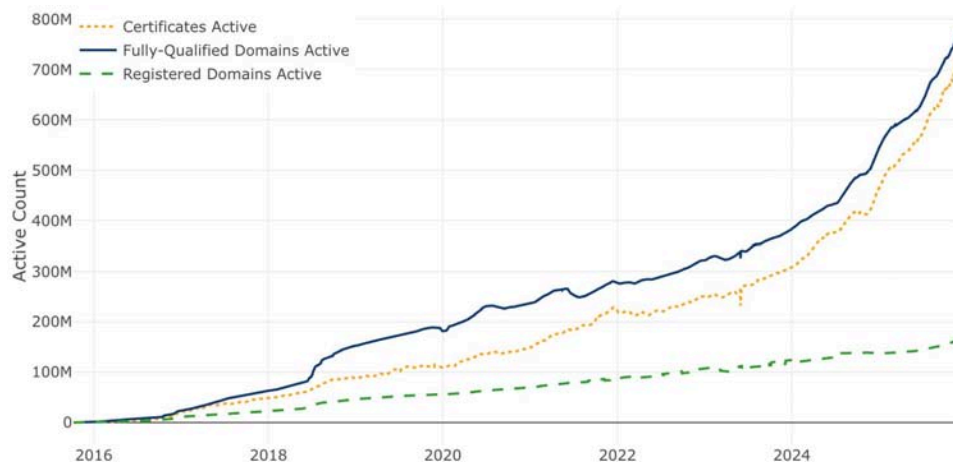
And speaking of the React vulnerability, something else happened that we've seen before when something bad was found in a significant piece of open source code: The pile on by security researchers all wanting to take a look turned up additional previously unknown problems. In this case, Meta, React's original creator and chief maintainer, has released new security updates for the React JavaScript framework. The new patches fix two denial of service bugs and a vulnerability that can expose an app's source code. So there is a bit of silver lining for the otherwise devastating React vulnerability. One thing we know is that motivating responsible security researchers to examine code is a terrific way to improve it.

Apple moves iOS to 26.2

Last Friday, Apple moved iOS to 26.2 which patched two actively exploited 0-day vulnerabilities in Webkit. Apple stated that the 0-days were used in an "extremely sophisticated attack" and that the targeted users were still running iOS versions earlier than iOS 26. Whether they, too, dislike Liquid Glass or may have older devices that cannot become current is unknown. You would hope that any target against whom it was worth launching a "extremely sophisticated attack" would be someone who understood that upgrading older hardware that can run the latest protections, even if it means somehow tolerating Apple's way-over-the-top UI nonsense, is still always the best policy.

Let's Encrypt to reach 1 Billion certs!

Let's Encrypt will cross a significant milestone in 2026, next year. With traditional certificate authorities establishing increasingly stringent security requirements to avoid spoofing, and with the coming ridiculous short lifetime certificates that will be putting a practical end to manual web certificate management, the lure of simply obtaining a domain validation certificate by providing proof of domain control through a DNS lookup and an ACME server answering at port 80 of the domains's IP ... well, that solution was always going to win. And winning it is:



Early last week, Josh Aas over at Let's Encrypt posted: "10 Years of Let's Encrypt Certificates" and he wrote:

On September 14, 2015, our first publicly-trusted certificate went live. We were proud that we had issued a certificate that a significant majority of clients could accept, and had done it using automated software. Of course, in retrospect this was just the first of billions of certificates.

Today, Let's Encrypt is the largest certificate authority in the world in terms of certificates issued, the ACME protocol we helped create and standardize is integrated throughout the server ecosystem, and we've become a household name among system administrators.

In 2023, we marked the tenth anniversary of the creation of our nonprofit, Internet Security Research Group (ISRG), which continues to host Let's Encrypt and other public benefit infrastructure projects.

Now, in honor of the tenth anniversary of Let's Encrypt's public certificate issuance and the start of the general availability of our services, we're looking back at a few milestones and factors that contributed to our success. A conspicuous part of Let's Encrypt's history is how thoroughly our vision of scalability through automation has succeeded.

In March 2016, we issued our one millionth certificate.

Just two years later, in September 2018, we were issuing a million certificates every day.

In 2020 we reached a billion total certificates issued ... and as of late 2025 we're frequently issuing ten million certificates per day.

We are now on track to reach a billion active sites, probably sometime in the coming year.

Think of that. One billion domain names. One billion certificates being continuously created, installed and replaced on web servers across the world. That really is an accomplishment.

As our listeners know, I've been a big fan and user of DigiCert's certificates ever since I left Verisign (who was later purchased by DigiCert). But this steadily shortening certificate life means that within another year or two I'll be joining the teeming billions whose certificates all say "Let's Encrypt." It's certainly no longer the case that Let's Encrypt certificates are in any way second class.

The browsers first decided to deprecate any extra value of cache' provided by extended validation (EV) certificates. So there was no reason to pay anything extra for them, and generic "Domain Validation" DV certs became the norm. Then the CA/Browser forum decided to abandon reasonably long-lived certificates just as Mozilla's efforts to solve the certificate revocation problem, offering total privacy on the client-side using Bloom filters, was finally working beautifully.

Oh well. As we've seen, the people who are driving the decisions behind technology do not always arrive at the best solution. But we can all celebrate Let's Encrypt's achievement.

I still shudder, however, at the idea that a billion websites are all now dependent upon a single service for their certificates and that if anything should happen to that service, websites will begin dropping off the air as they become unable to obtain the required certificate renewal.

The genius of the Internet's design has always been its distributed diversity without any single point of failure. This changes that. I hope we know what we're doing.

DNS Benchmark Update

I'm pleased with the first 10 days of the DNS Benchmark's wider audience response. We're now at its 3rd release which new purchasers will receive, and everyone who runs either of the first two releases, unless they have deliberately turned off its check for updates, will receive a notice that a newer release is available, along with a link to download the latest release.

The primary incentive for the 2nd release was the discovery that I had not allocated large enough string buffers in the code that posts the conclusions to contain the total number of packets sent and received. When the benchmark's 50x or 100x sampling modes was chosen, more than one million queries might be sent. That was 50 or 100 times more than the benchmark ever used before. And I had failed to increase the buffers to contain the resulting larger strings. So the second release fixed that along with some random cosmetic things.

The primary incentive for the 3rd release was the discovery that we needed at least version 9 of the Windows WINE emulator. A surprising number of people are running the DNS benchmark under Linux and Mac, and despite the fact that WINE 9 is nearly two years old and 10 was released at the start of this year, many people still have WINE 8. The problem is that WINE 8 predated the change in code signing from SHA1 to SHA256. The DNS Benchmark verifies its own digital signature at startup to make sure that it was properly downloaded and saved. So that was failing for those who were still using WINE 8. But the problem was, under the assumption that the only reason for a signature failure would be code modification, the error message that was being presented was confusing, stating that something must have been altered the program after it was downloaded. So I quickly pushed the 3rd release out to end the confusion about WINE 8. Now the Benchmark first checks to see whether it's running in WINE and, if so, whether it's WINE 9 or later. If it's WINE 8 or earlier it explains that the user will need to update to a later version of WINE. One other improvement was someone asking to have Ctrl-C copy the text of any of the program's many dialogs to aid in translation. So that's now there, too.

We've received a bunch of gratifying feedback. Some from people who cannot get their heads around how much is packed into a 215 kbyte Windows program – everything it does and all of the many descriptive dialog windows it contains. We have all been so abused by the ridiculous multi-100 megabyte monstrosities that we've lost sight of how dense and expressive code can be.

The other feedback has been, as predicted, from people who are commenting that they were happy with their local DNS resolver whose caching performance was insanely high. The problem with having an insanely high performance local DNS router cache is that its contents will be largely the same as the DNS cache in Windows itself because the request from Windows will be what caused the router to load its cache. So Windows won't ask again for anything that's cached since it will already have it. Since version 1 of the DNS Benchmark placed cached performance above all else, anyone local router was always ranked #1. Not so for the much improved version 2. So this has been a bit of a wake up call for those people. But I'm glad for the changed strategy which I think more accurately reflects what's needed to make today's widely sourced websites fly.

Listener Feedback

Scott Wise

Steve, An issue I see with Age Grouping or the OverAgeTokenCredential for age verification that I'm sure you've thought about, but I don't remember hearing discussed, is that it will disclose your birthday on your birthday:

If you need to be a certain age to access a service, be it physically or virtually, you likely do it on, or very near, your birthday. A common one is going drinking on your birthday when you are exactly old enough. In the physical realm, you'll likely get a "congratulations" and maybe even a free drink, but they don't share your personal information with others.

If you need to be a certain age to access social media, you are likely to create an account on your birthday and you should be assured that the company will sell that information to as many others as they can.

Reaching certain ages will trigger different ads. Driving age will likely trigger car sales ads and reaching the drinking age may trigger alcohol ads. These will happen regardless of your actual birthday as they would fall into the Age Group identification. I don't know all the ramifications of disclosing your birthday, but a few I can think of would be enhanced phishing, fake account creation and password guessing. This isn't a reason to stop the work on Age Grouping or the OverAgeTokenCredential, but I think it should be considered.

Scott / Regina, Saskatchewan, Canada

I think Scott makes a terrific point. In a world where accounts on highly desirable services are age-gated, the first-time start of use of those services could reasonably be used to infer something about the individual's age. Back in the '70s, though apparently less so today, knowing when someone had obtained their driver's license would, with some accuracy, tag their age. Though today's teenagers seem to feel less urgency to drive, that urgency to drive may have been replaced with an urgency to use social media. If we imagine a world where we have robustly solved the online proof-of-age problem, it's easily foreseeable that anyone turning 16 in Australia, and soon in many other jurisdictions, would immediately join the many services that they would then be able to on the day of their 16th birthday. So, as Scott points out, that constitutes a strong disclosure of age.

Mr. Gecko

There is one problem with the age verification solution that is being worked on, and that is the discrimination of what device and operating system one must use to be considered valid. The attestation system discriminates against open source operating systems, browsers, and even prevents new competition from being able to start. This means to use the internet, social media, or anything considered adult content... One must use an Apple or Microsoft based computer, with an Apple, Microsoft, or Google browser. And one must use iOS with Safari, or one of the approved Android phone vendors with original software. This will be very very bad, and no one is talking about the issue from this standpoint.

I personally install an open source operating system on both my phone and PC to get away from the privacy invading companies. Once these laws come into play, I won't be able to use Facebook to contact my parents (who would not use signal or any other messaging solution), and will be treated like a bad guy because I decided to go for privacy.

This is another great point. Our Mr. Gecko is noting that the requirement of bringing enforceable security to age verification means that platforms which are unable to offer true enforceability will not be permitted to assert their user's age. And as listeners of this past know, this is a common theme. Hundreds of millions of otherwise completely functional PCs are stuck at Windows 10 because they only contain hardware or firmware support for version 1.2 of the TPM and Microsoft has decided to require TPM 2.0 for Windows 11 and beyond. Another example is the defacto requirement that Windows executables be signed with an expensive cryptographic certificate that expires every few years for no real reason other than to create revenue for certificate authorities. As Mr. Gecko noted, all of this is hostile to open source, open software and open platforms. All security absolutely requires the ability to robustly keep secrets of some kind somewhere. Yet full openness is explicitly about never keeping secrets. The two concepts are fundamentally at odds with one another.

Owen LeGare

Hi Steve, I have a question regarding the grc.sc/botcheck shortcut you created. In the podcast discussion of the service, I don't recall any mention about when you get a result showing activity, how to determine if that activity is from your network or whoever had been assigned that IP address previously. Since most people get their IP address by DHCP, the activity could be from someone who had the IP previously.

If there is a way to determine how long you have had your IP address and the botcheck site shows the dates when the malicious activity occurred, you should be able to determine if all the activity was before you were assigned the IP address. Is this the way to make that determination? / Owen

Owen makes an extremely good point. For those whose IP addresses change often, this test could be inaccurate in both directions: It could produce false-positives or false-negatives by reporting on the condition of the network of whomever one or more people may have had that IP previously. The other problem is that IPv4 depletion has moved some large ISP hosts to Carrier-Grade NAT. When an ISP has more subscribers than they have IPv4 addresses, and when they are unwilling or unable to upgrade their services to IPv6, they will be forced to place NAT routers between their subscribers and the Internet – just as we end-user subscribers have many more Internet gadgets than we have public IP addresses. My point is that Carrier-Grade NAT, which is becoming increasingly common, will also obscure the truth since any one of the ISPs subscribers may have been emitting malicious Bot traffic from the public IP that is now assigned to the user running the botcheck service.

So it's true that all of those caveats need to be taken into consideration when using that free botcheck service. My IPs with COX Communications and my cable modems remain static for years at a time. So for those whose public IPs are also not changing the boxcheck service can be useful.

Allen W. (we've heard from Allen, our voracious SecurityNow-consuming semi-truck driver)

Steve,

I last wrote to you at the end of October asking about a password of 63 plus signs. I just finished that episode and you're right, I understand now. Kinda cool that I wasn't far off in my assumption of 63 plus signs being a strong enough password. Great episode - thank you. Yes, I have listened up to episode 303 since late October and not just during my 70-hour work week driving a semi. I've found myself spending a large portion of my waking hours listening

to Security Now.

I was thrilled when you read my last email on Security Now and, yes, my Python sensei Shawn did indeed share that clip with me. As you mentioned, by the time I get caught up, I'll be a completely different person. I'm seeing that already.

As you can imagine, listening to 50+ hours of Security Now per week while driving a semi (and then more listening after hours) has made me quite paranoid about everything security and it's constantly on my mind. Perhaps my brain is in overload.

I took the week of Thanksgiving off and didn't listen to a single episode for 6 days. I felt less nervous about security two days into my break but then during a train ride the strangers at the table with me started talking about loving their debit card's tap feature and moments later I found myself lecturing everyone about what could happen thanks to that little chip. Later I realized that most of what I said was probably based on information from before Michael Jackson died but even in hindsight, no regrets. They got off light since I didn't make them buy a copy of SpinRite before disembarking the train.

Since listening to episode 303, I'm going to ask about something you've mentioned a few times: I understand that every successive binary bit represents a doubling of values, but I've also heard you say that with 26 letters in the alphabet, double that for upper and lower case, and add numeric digits, that would give 62 possible combinations out of 64 total in a 6 bit word. I've heard you run the math which reveals 5.9375 bits of entropy – just shy of 6 bits. Considering that binary is either a 0 or a 1, how is that the .9375 is not rounded up to 1? I'd think someone trying to brute force the number would have to try all 64 combinations. Wouldn't just letters of a single-case, plus numbers, giving 36 possible combinations, take the same amount of time to crack, since all 6 bits would have to be tested? The brute forcing system wouldn't know to just test the first 36 of 64 values, right?

Thank you and Leo for this podcast! Sitting in traffic is a lot less rage inducing since I started listening and that's a good trade for the cold sweats I get until my VPN reconnects every time I reboot my computers or cell. / Allen

The answer to Allen's question is that, in fact, the effective entropy really is that odd-seeming 5.9375 bits of entropy because lower-case plus upper-case plus 10 digits create, in this example, a total of 62 characters in this reduced alphabet. And even though expressing any one of those 62 characters in our reduced alphabet does require 6 bits, there *is* no character represented by the 63rd and 64th binary bit patterns. Those final two binary bit patterns do not stand for anything. They do not represent any character of our reduced alphabet so they actually cannot be tested. We must stop after testing the 62nd character – which is at the end of our alphabet – reset it back to zero, increment the next most significant character to its next possibility, and keep trying.

Louis Blanchard

Hello Steve, I continue to really enjoy the Security Now! podcast.

*I recently purchased a new **MikroTik hEX S (2025)** router (running firmware **RouterOS 7.20.6**), which offers excellent value. After updating the device and installing it in my home network, I ran the **ShieldsUP!!** test to assess its default security posture.*

*I was pleased to see that the "All Service Ports" check reported "**Stealth**" for all open TCP ports. However, ShieldsUP!! was able to elicit a reply to an **ICMP Echo Request (Ping)**. I confirmed this behavior after a factory reset, indicating it is the device's default configuration. I have since configured the firewall to drop inbound ICMP Echo requests, resolving the issue. My question is about the security implications of this default setting:*

- *Is shipping a router with default ICMP Echo replies enabled potentially negligent or dangerous for general customers who may have little networking knowledge?*
- *Given that MikroTik often uses the same default configuration across many hardware models, would it be worthwhile to contact MikroTik to suggest a change to the default firewall template to drop WAN-side ICMP Echo requests? (While ensuring vital ICMP traffic, such as Destination Unreachable, remains active for performance.)*

Best wishes to you and your family for the holidays! / Thanks, in advance Louis

This is a great question. It's one of those issues, like the undeniable utility of NAT routing, that causes the old greybeard Internet Unix gurus to increase their blood pressure medication. The reason for this is that it is absolutely clear that any IP device that's alive and working should, at the absolute minimum, reply to an ICMP echo request with an ICMP echo reply. If an IP protocol stack is present and connected the specifications are very clear that this should be done. The argument could be made, and believe me, the cranky old greybeard Internet Unix gurus do so, that any device that deliberately fails to do this simplest of all things is an aberrant abomination on the Internet, has no right to send or receive a single IP packet, and should be immediately disconnected — with prejudice — and burned at the stake.

I completely understand what those people are saying. And they're not wrong. ICMP echo requests and replies, commonly referred to as "pings" are incredibly useful. They are perhaps one of the most useful features of IP networking. By being so low level, by not relying upon anything else to function, by – by default – always being present, it's possible to "ping" any device at any IP and to know that you'll receive a reply if that device is alive and if IP traffic has managed to get to and from the source of the ping and its destination.

So, in a very real sense, deliberately NOT replying to a ping request – just ignoring it – is a breach of one of the most fundamental laws of the Internet Protocol. The flip side is to ask who is pinging us and why? Would we want a tech who works for our ISP to be able to ping our router if they are diagnosing some network trouble? Yeah, we probably would. But would we want to reply to an ICMP Echo Request from some random hacker in North Korea, China or Russia? How does telling them that "Hey, yeah, we're here! What did you have in mind?" possibly help us?

The problem with those old greybearded Internet Unix gurus is that they're living in their own ivory tower. They'll say "Well, of course you should have a good firewall!" Right. But what if that firewall contains a known bug that requires a bunch of pounding on its wall to penetrate? No one is going to bother pursuing a difficult-to-exploit vulnerability against an IP that doesn't reply and may just be dead air. But if that same IP bounces back a "Hiya! What's up?" response to anyone anywhere in the world who knocks? You just might find yourself on the receiving end of an attempt to penetrate your defenses.

I'm not saying that any of that is likely to happen, but it's a valid scenario. I think the question to ask oneself is how it benefits YOU to have the device that's protecting your entire network announcing its presence to anyone anywhere who attempts to bounce a "ping" off of its public interface? If running with full "stealth" is an option, I don't see any reason not to use it. And if you **are** working with your ISP's tech, I'll bet they know by now to ask you to disable your router's "Stealth Mode" if they are using ICMP Echo Requests to troubleshoot your connection to their network.

Australia

I expect to be giving this entire age-verification issue a rest for a while, at least until something more happens. But before doing that I wanted to wrap up today's podcast with a check-in on the status of Australia's social media age restrictions, some comments from two of our listeners, and my clearest-yet description of where we should and where we should not compromise.

So what's going on in Australia? To say that the entire world is watching with interest would be no exaggeration. You would think that the world's news reporting agencies were starved for news with all of the coverage this ban has been attracting. Everyone is watching and everyone is reporting.

Sadly, "technology" in general is not showing too well. "Technology's" reputation is taking a bit of a beating because Australia's teens are being confronted with age detection based on facial feature characteristics which everyone knows to be readily spoofable, and no one is being disabused of that belief. There are stories of girls applying more makeup to appear older and slipping right past the detector. Or a 13-year old boy who scrunched-up his face when asked to verify his age. Presumably, the scrunched-up face looked old, wrinkled and prune-like. That's all it took. Other teens have simply had someone older look into the camera for them. And on the flip side, 16, 17 and 18 year olds have been banned for being underage. So this is not technology's proudest time.

I've seen stories of parents who, for whatever reason, believe in raising their children to be their best friends. They believe that over-exposure to social media may not be healthy for their kids, but remaining their child's best friend means that someone else needs to tell them 'no'. So these parents have been disappointed when their 12-year old was accepted as being 16 and allowed to continue using social media. They were hoping it would end... but what can they do? Right. At the same time, there have been stories of teens expressing relief at being denied and blocked due to their age. If they could be involved in the social media rate race, they needed to be. But they are not unhappy to now be off the hook, at least for a few years.

Maybe the practice of facial age-spoofing will just become another game the kids play that shows how dumb the adults are. At this moment it would be difficult to argue with that. My advice to the facial detection providers would be to invest the profits being enjoyed today, because I'll bet they're going to be short lived. It's bad that facial age detection is such an inherently inexact practice. I completely understand that this is all we have right now, but it has been misapplied for this application. It should never have been used here, since whether or not teens are old enough to have access to the social media which is often central to their lives – independent of whether or not this might be healthy – cannot be left to chance and to a capricious utterly error-prone technology. My point is, the go/no-go decision is too important and must be made fairly and based upon an individual's actual age, not a coin toss. Some are saying that anything is better than nothing, but I would disagree.

Among the rest of the world that's watching this first-ever nationwide experiment is the EU. They may be further along with an application that can verify someone's age without any privacy compromise. One thing is for sure: Anyone who may have believed that facial feature age determination actually works well enough probably no longer thinks so.

I encountered a note from an Australian listener that I felt provided a valuable perspective.

Bruce French wrote:

Hi Steve and Leo, I am a long time listener (and Club TWIT member) here in Adelaide, Australia. (I have been listening to Leo since he was in the Cottage). I have been listening to your discussion on the Social Media Ban (SN Episodes 1054 & 1055) just implemented here and thought it may be useful to put my point of view and (limited) experience forward.

Firstly, this small part of the world has not stopped functioning. It just has not been a huge deal! There has been the usual commentary on the various media outlets; some teens have been able to bypass the restrictions, some adults have been blocked when they should not have. BUT, these do not appear to be in large numbers; no mass commentary in either direction. Nobody within my extended family has been asked to verify their age. Unless you are under 16 or near to 16 years old, it has been a non event for the majority.

When listening to episode 1054 I became a little defensive (of Australia I suppose) as I thought your tone was a little condescending and mocking. I think I understand more where you are coming from after just listening to SN1055, privacy being the issue of concern.

Right. And as I noted, fairness and correctness. A flaky age verifier seems like a bad thing.

I would like to make the point that this Social Media ban is essentially driven by the people, just implemented by the government. There have been a number of well documented cases here where teens have died (suicide) as a result of Social Media abuse. With the Media companies refusing to control/take-down the relevant content the public has essentially asked the government to do something.

There looks to be strong majority support throughout the country for this action. I have no data on hand to support this, but it is clear there has been very little pushback other than from the Media companies.

I note here that in general in Australia we are prepared to have some restrictions imposed if it is for the greater good, this looks to be one of those times. There even appears to be support (though begrudging) from a good proportion of the teens affected.

Of course all of this is from my very limited viewpoint, (I do not have teenagers myself and my grandchildren are not yet old enough for this to have been an issue). Regards, Bruce French

It's interesting that Bruce reports that adults are not being asked to verify their ages. There are presumably other heuristics that the services could be using. I encountered an interesting thought somewhere, which noted that anyone having an existing account that's at least 10 years old could safely be assumed to be at least 16 years old since they would have had to create the account ten years ago when they were younger than 6, which seems unlikely.

And I agree with Bruce from my casual survey of the reporting. I did get the general sense that only some younger teens are chaffing. Many of their peers seem somewhat relieved that peer pressures are no longer forcing their reluctant participation and the sense of relief among the adult and parent population appears to be nearly universal.

Next, I wanted to share and react at length to what another of our listeners wrote.

Jane:

Greetings Steve! In the previous episode, as well as in a number of previous ones, you expressed an opinion that a universal, location-independent age verification standard should be developed, as it is the direction the Web is going. However, I find it unsettling when this idea is treated as an acceptable compromise.

The "privacy preservation" in at least some of the methods discussed (such as the one described in Episode 1044) can be negated by subpoenas (as admitted in that episode as well), thus leaving people still very vulnerable. This could also be a very convenient avenue to also deny certain adults (like journalists) access to various resources based on political reasons.

One particular approach was mentioned, and it was Apple or Google doing the verification, maybe on-device. Even if we assume no logs are sent out to their servers, this is a very damaging solution. Right now switching to a freedom-respecting OS is more important than ever, with the increasing surveillance, attacks on freedom and just plain enshittification (like the proposed limitations on unverified APK installation). After having used Graphene for 1.5 years now, I don't think I can go back to having Google services at all, let alone with maximum privileges. Same for Linux on my desktop.

In the newest episode, it was phrased as "we can do it without any loss of privacy". But the biggest loss of privacy in the proposed solution - in addition to surrendering your sensitive documents to Apple/Google - is the loss of privacy on the device itself, by having to have the invasive Google services privileged in the system. Same goes for Apple, except there aren't de-Apple'd OS versions in the first place. A whole other problem in itself is having to register with and agree to the terms of a third party completely unrelated to the service one was going to use.

De-googled Oses are already being disadvantaged - banking and other important apps would often refuse to run on non-stock Oses, which is one of the biggest hurdles to adoption. You can get around this with root and certain tools, but that's apparently becoming harder, doesn't always work and is a continuous fight, not to mention that would mean still having the invasive Google services installed. Age confirmation is likely to be treated as strictly as identity documents or banking - thus effectively excluding people like me. There is an example of this already - the European identity wallet, which was mentioned in this podcast, has been found to employ Play Integrity. An issue was raised on Github (<https://github.com/eu-digital-identity-wallet/eudi-app-android-wallet-ui/issues/287>) but at least last I checked, the developers dismissed it.

I find it odd to treat widespread age verification as any less horrific than Chat Control. This would cause just as much collateral damage, if not more, far outweighing any potential benefits (and it'd be unlikely to "protect children" anyway).

Thanks for the podcast, it's one of the reasons I switched my education direction towards security!

I want to first note one point: Jane notes that the "privacy preservation" of the system we talked about in Episode 1044 could be negated by subpoenas. To be very clear, I would never consider any system whose privacy can be breached by a court order to be sufficiently privacy preserving. I mentioned that aspect of the TruAge system specifically because that was a red flag. We know how true privacy fanatics such as Apple and Signal would respond. They would have deliberately designed their technology so that they are technically unable to respond to any court order.

But the bigger point I want to make here reflects Jane's very understandable absolutist attitude which she shares with many of our listeners. The Internet has been commercialized and its users are being monetized, whether we like it or not. The commercial interests such as Apple and Google have grown into monopolies that no longer have any meaningful competition, and the various governments of the world are unhappy that the Internet fights against their desire to monitor and control what their citizens – and even other country's citizens – are allowed to do.

Jane started out her note writing: *"In the previous episode, as well as in a number of previous ones, you expressed an opinion that a universal, location-independent age verification standard should be developed, as it is the direction the Web is going. However, I find it unsettling when this idea is treated as an acceptable compromise."* I understand what Jane means. But in Australia today young people are being forced to stare into a camera's lens so that their image can be transmitted to some third-party service to judge their age. That's not privacy preserving by anyone's standard.

The question is no longer whether or not Internet users are going to be able to continue to enjoy completely unfettered access to any resource anywhere they choose. That's what's known as a lost cause. Those days are over. Our various governments are taking those days away. So it's not about having "acceptable compromises". Whether we like it or not, with or without us, Control is descending upon the Internet.

Apple already knows all about me. I subscribe to Apple TV and News and have ApplePay setup in my iPhone. So I have no problem with the idea that Apple would allow my smartphone to assert my age – and absolutely nothing else – to anyone who has a need to know, after I've given my permission. I can't say that I trust Google to the same extent. But perhaps the Android platform will find a savior to offer universally accepted age assertion.

What's possible from a pure technology standpoint – and this is where "acceptable compromise" comes in for me, but also where I see no reason to further compromise beyond that, at all, ever – is for individuals to affirmatively identify themselves just once to one trusted proxy under the understanding that while that proxy must briefly know who they actually are in the physical world in order to verify their date of birth, that proxy will then discard all of that transient identifying information, retaining only their date of birth and the information required to identify them biometrically from then on.

From that point forward, at any time, they can call upon that proxy to present to any inquiring third party an anonymous assertion of whatever age is required. If we can get that, it will be a lot. It should be the industry's goal. I'm seriously annoyed that Apple has not yet stepped up with the realization that it is in the best interests of their users for their iDevices to be able to make those assertions on their behalf. Apple should be the one to do this. They are hurting the privacy of their users by continuing to refuse. None of the children who are staring into an iPhone in Australia should need to be scrunching up their faces or applying makeup and having their photos sent to third party services. Not when Apple could entirely solve this problem without breaking a sweat.

The noises the EU is making along these lines all sound right. And there's talk of some app that's presumably EU-centric and cross platform, so that both Apple and Android would be covered. That may be where the rest of the world is looking next.

