Security Now! #1019 - 04-01-25
EU OS

This week on Security Now!

- Kuala Lumpur International Airport says no to a ransom attack, switches to whiteboard.
- A tired and jet-lagged Troy Hunt got Phished then listed himself on his own site.
- Cloudflare completely pulls the plug on port 80 (HTTP) API access.
- Malware is switching to obscure languages to avoid detection. FORTH, anyone?
- Password reuse doesn't appear to be dropping. Cloudflare has numbers.
- A listener shares his log of malicious Microsoft login attempts. Why no geofencing?
- 23andMe down for the count (reminder).
- A sobering Ransomware attack & victim listing website. Gulp!
- "InControl" keeps VR planes aloft.
- And the European Union gets serious about a switch to Linux.

Subtle coding choices can land you at the bottom of the canyon

Security News

Malaysia declines to pay ransom after an airport attack
Malaysian Prime Minister Anwar Ibrahim has declined to pay a $10 million ransom after hackers paralyzed IT systems at the country's main airport over the weekend. The incident forced staff at the Kuala Lumpur International Airport to manually post flight information on a large whiteboard with a pen:
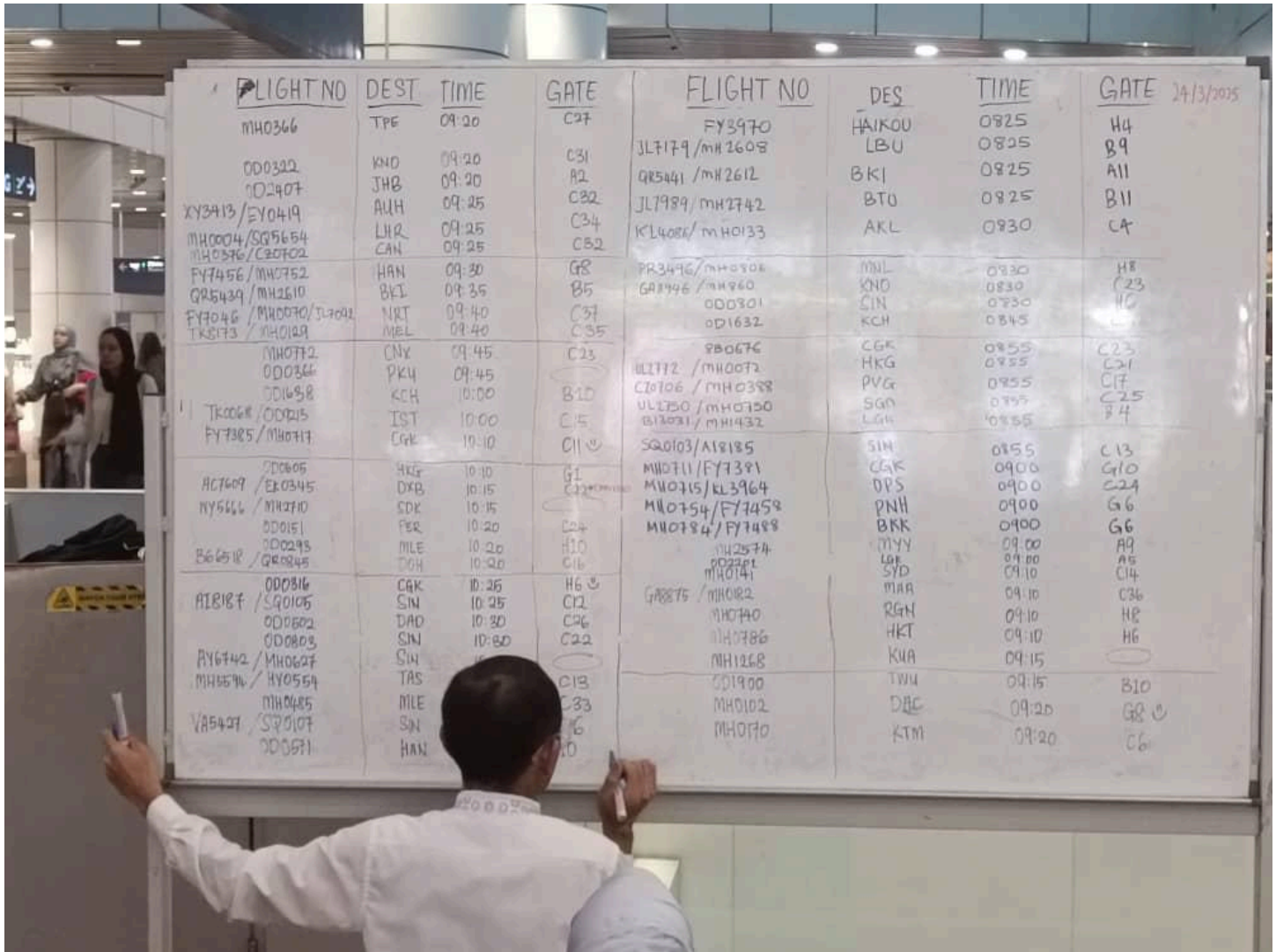

Malaysia's Prime Minister said it took him less than 5 seconds to decide to decline to pay the ransom. So far, no specific group has taken credit for the attack so far.

This story would not usually have risen to the level of news for today's podcast, but the accompanying photo, showing the whiteboard that was deployed to replace the automated plane schedules and departure gates, was too wonderful and poignant to pass up.

Troy Hunt got Phished
We would file this one under the heading *"It can happen to the best of us."* Or perhaps *"Even the most security-aware person can trip up."* Last week, Troy Hunt who's famous for his "Have I been Pwned" password leakage tracking site and service posted his piece titled *"A Sneaky Phish Just Grabbed my Mailchimp Mailing List"*. Troy wrote:

You know when you're really jet lagged and really tired and the cogs in your head are just moving that little bit too slow? That's me right now, and the penny has just dropped that a Mailchimp phish has grabbed my credentials, logged into my account and exported the mailing list for this blog. I'm deliberately keeping this post very succinct to ensure the message goes out to my impacted subscribers ASAP, then I'll update the post with more details. But as a quick summary, I woke up in London this morning to the following:
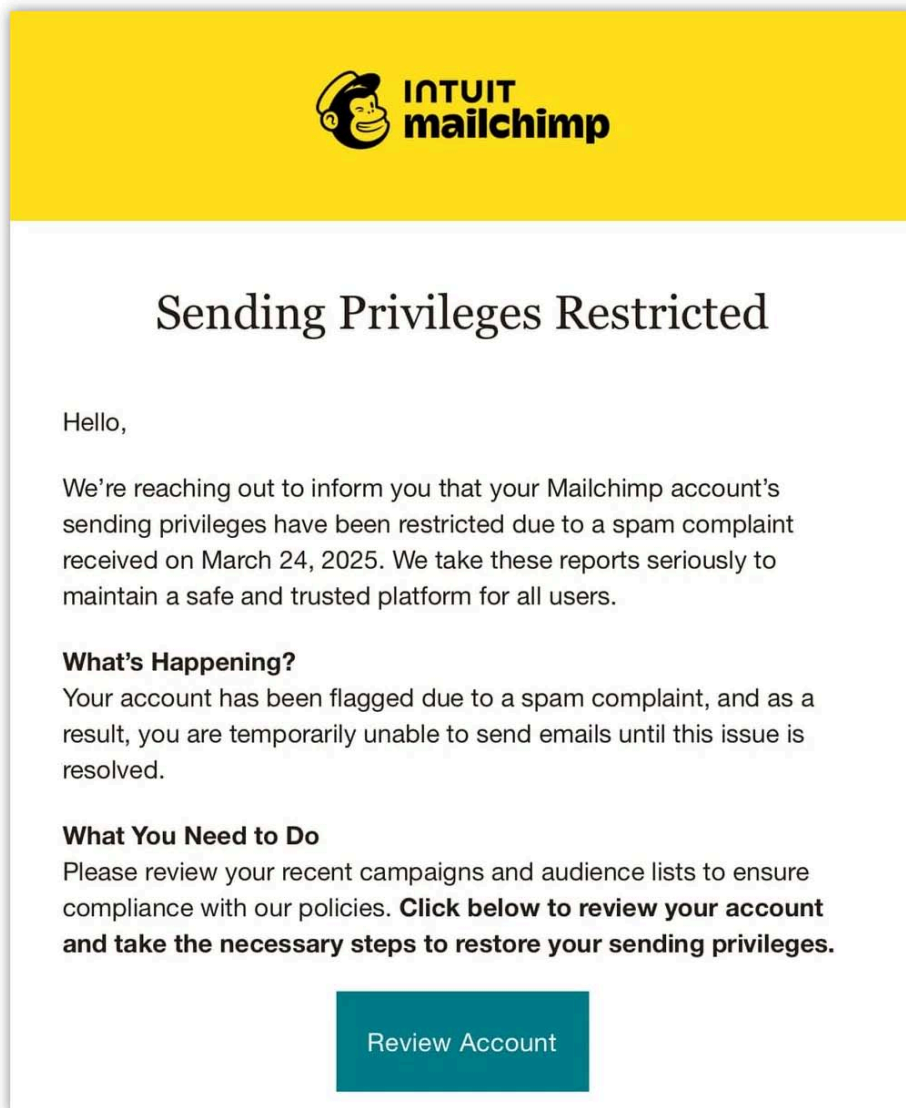
I went to the link which is on mailchimp-sso.com and entered my credentials which - crucially - did not auto-complete from 1Password. I then entered the OTP and the page hung.

Moments later, the penny dropped, and I logged onto the official website, which Mailchimp confirmed via a notification email which showed my London IP address. I immediately changed my password, but not before I got an alert about my mailing list being exported from an IP address in New York. And, moments after that, the login alert from the same IP "We'd like to confirm some recent activity on your account."

This was obviously highly automated and designed to immediately export the list before the victim could take preventative measures. There are approximately 16k records in that export containing info Mailchimp automatically collects.

Every active subscriber on my list will shortly receive an email notification by virtue of this blog post going out. Unfortunately, the export also includes people who've unsubscribed (why does Mailchimp keep these?!) so I'll need to work out how to handle those ones separately. I've been in touch with Mailchimp but don't have a reply yet, I'll update this post with more info when I have it.

I'm enormously frustrated with myself for having fallen for this, and I apologise to anyone on that list. Obviously, watch out for spam or further phishes and check back here or via the social channels in the nav bar above for more. Ironically, I'm in London visiting government partners, and I spent a couple of hours with the National Cyber Security Centre yesterday talking about how we can better promote passkeys, in part due to their phishing-resistant nature. 🤷‍♂️

More soon, I've hit the publish button on this 34 mins after the time stamp in that first email above.

So that was the blog posting that he quickly pushed out to let his more than 16 thousand subscribers know that the email address they share with him had escaped. Later Troy continued under the headline *"More Stuff From After Initial Publish"*:

Every Monday morning when I'm at home, I head into a radio studio and do a segment on scams. It's consumer-facing so we're talking to the "normies" and whenever someone calls in and talks about being caught in the scam, the sentiment is the same: "I feel so stupid". That, friends, is me right now. Beyond acknowledging my own foolishness, let me proceed with some more thoughts:

Firstly, I've received a gazillion similar phishes before that I've identified early, so what was different about this one? Tiredness was a major factor. I wasn't alert enough, and I didn't properly think through what I was doing. The attacker had no way of knowing that (I don't have any reason to suspect this was targeted specifically at me), but we all have moments of weakness and if the phish event is timed perfectly with that, well, here we are.

Secondly, reading it again now, that's a very well-crafted phish. It socially engineered me into believing I wouldn't be able to send out my newsletter so it triggered "fear", but it wasn't all bells and whistles about something terrible happening if I didn't take immediate action. It created just the right amount of urgency without being over the top.
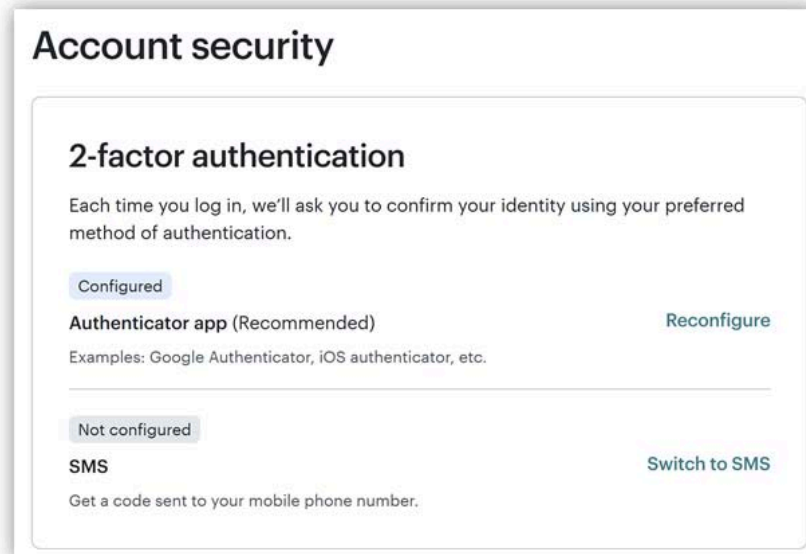
*Thirdly, the thing that **should** have saved my bacon was the credentials not auto-filling from 1Password, so why didn't I stop there? Because that's not unusual. There are so many services where you've registered on one domain (and that address is stored in 1Password), then you legitimately log on to a different domain. For example, Qantas airlines uses both "www.qantas.com.au" and elsewhere "accounts.qantas.com".*

And the final thought for now is more a frustration that Mailchimp didn't automatically delete the data of people who unsubscribed. There are 7,535 email addresses on that list which is nearly half of all addresses in that export. I need to go through the account settings and see if this was simply a setting I hadn't toggled or something similar, but the inclusion of those addresses was obviously completely unnecessary. I also don't know why IP addresses were captured or how the latitude and longitude are calculated. But given I've never seen a prompt for access to the GPS, I imagine it's probably derived from the IP.

> *I'll park this here and do a deeper technical dive later today that addresses some of the issues I've raised above.*

And a bit later Troy continued:

> *Unfortunately, Mailchimp doesn't offer phishing-resistant 2FA:*
>
> **Account security**
>
> **2-factor authentication**
> Each time you log in, we'll ask you to confirm your identity using your preferred method of authentication.
>
> Configured
>
> **Authenticator app (Recommended)**   Reconfigure
> Examples: Google Authenticator, iOS authenticator, etc.
>
> Not configured
>
> **SMS**   Switch to SMS
> Get a code sent to your mobile phone number.
>
> *By no means would I encourage people not to enable 2FA via OTP, but let this be a lesson as to how completely useless it is against an automated phishing attack that can simply relay the OTP as soon as it's entered.*

Troy also wrote: *"I just went to go and check on the phishing site with the expectation of submitting it to Google Safe Browsing, but it looks like that will no longer be necessary"* because he was presented with a Cloudflare intercept page stated that the page was suspected of being used for phishing. Troy added:

> *2 hours and 15 minutes after it snared my creds, Cloudflare has killed the site. I did see a Cloudflare anti-automation widget on the phishing page when it first loaded and later wondered if that was fake or they were genuinely fronting the page, but I guess that question is now answered. I know there'll be calls of "why didn't Cloudflare block this when it was first set up", but I maintain (as I have before in their defence), that it's enormously hard to do that based on domain or page structure alone without creating a heap of false positives.*

Troy knew that he would need to load those addresses into his own "Have I been Pwned" site. He wrote:

> *When I have conversations with breached companies, my messaging is crystal clear: be transparent and expeditious in your reporting of the incident and prioritise communicating with your customers. Me doing anything less than that would be hypocritical, including how I then handle the data from the breach, namely adding it to HIBP. As such, I've now loaded the breach and notifications are going out to 6.6k impacted individual subscribers and another*

https://www.troyhunt.com/a-sneaky-phish-just-grabbed-my-mailchimp-mailing-list/

I've included a link to Troy's entire blog posting which proceeds with, at the time of this writing, a series of seven follow-ups.

He spends a lot of time looking at the many benefits of Passkeys, which are inherently phishing resistant, because the information being sent back to the authenticating server is neither static username and password, nor short-duration one-time-codes. The authenticating server sends a unique never-before-seen challenge over an end-to-end encrypted link which the user's client signs. So any man in the middle is cut out.

But the biggest takeaway is that phishing, which takes advantage of the human factor, remains an active threat today; and it can literally happen to anyone – even someone as astute as Troy who lives and knows this stuff inside out. It just happened to catch him at a time of fatigue and jet-lagged weakness; but it did catch him.

The addition of one-time-passwords has neutered non-real-time attacks where a user's logon username and password have been stolen. But automated attacks which immediately forward the user's provided one-time-password to the authenticating server remain 100% effective. And we've also seen how ridiculously long some authenticators such as Microsoft will continue to honor a token that expired many many minutes earlier.

And the fact that the attackers used the domain "mailchimp-sso.com" further masked the attack, even to someone like Troy who probably noticed. That was a perfectly reasonable domain name of the sort we see everyday.

Cloudflare pulls the plug on port 80 API access
Their blog posting was titled *"HTTPS-only for Cloudflare APIs: shutting the door on cleartext traffic"*. They introduced the change by writing:

Connections made over cleartext HTTP ports risk exposing sensitive information because the data is transmitted unencrypted and can be intercepted by network intermediaries, such as ISPs, Wi-Fi hotspot providers, or malicious actors on the same network. It's common for servers to either redirect or return a 403 (Forbidden) response to close the HTTP connection and enforce the use of HTTPS by clients. However, by the time this occurs, it may be too late, because sensitive information, such as an API token, may have already been transmitted in cleartext in the initial client request. This data is exposed before the server has a chance to redirect the client or reject the connection.

A better approach is to refuse the underlying cleartext connection by closing the network ports used for plaintext HTTP, and that's exactly what we're going to do for our customers.

> *Today we're announcing that we're **closing all of the HTTP ports on api.cloudflare.com.** We're also making changes so that api.cloudflare.com can change IP addresses dynamically, in line with on-going efforts to decouple names from IP addresses, and reliably managing addresses in our authoritative DNS. This will enhance the agility and flexibility of our API endpoint management. Customers relying on static IP addresses for our API endpoints will be notified in advance to prevent any potential availability issues.*
>
> *In addition to taking this first step to secure Cloudflare API traffic, we'll provide the ability for customers to opt-in to safely disabling all HTTP port traffic for their websites on Cloudflare. We expect to make this free security feature available in the last quarter of 2025.*
>
> *We have consistently advocated for strong encryption standards to safeguard users' data and privacy online. As part of our ongoing commitment to enhancing Internet security, this blog post details our efforts to enforce HTTPS-only connections across our global network.*

https://blog.cloudflare.com/https-only-for-cloudflare-apis-shutting-the-door-on-cleartext-traffic/

Their posting then goes into great detail with networkstate diagrams and more, about how and why none of the options for redirecting initially plaintext HTTP traffic to HTTPS is able to achieve the same absolute level of security as simply saying 'no' to all non-HTTPS traffic from the start. They wrap up this posting by writing:

> *Starting today, any unencrypted connection to api.cloudflare.com will be completely rejected. Developers should not expect a 403 Forbidden response any longer for HTTP connections, as we will prevent the underlying connection to be established by closing the HTTP interface entirely. Only secure HTTPS connections will be allowed to be established.*
>
> *We are also making updates to transition api.cloudflare.com away from its static IP addresses in the future. As part of that change, we will be discontinuing support for non-SNI legacy clients for Cloudflare API specifically — currently, an average of just 0.55% of TLS connections to the Cloudflare API do not include an SNI value. These non-SNI connections are initiated by a small number of accounts. We are committed to coordinating this transition and will work closely with the affected customers before implementing the change. This initiative aligns with our goal of enhancing the agility and reliability of our API endpoints.*
>
> *Beyond the Cloudflare API use case, we're also exploring other areas where it's safe to close plaintext traffic ports. While the long tail of unencrypted traffic may persist for a while, it shouldn't be forced on every site. In the meantime, a small step like this can allow us to have a big impact in helping make a better Internet, and we are working hard to reliably bring this feature to your domains. We believe security should be free for all!*

This is the sort of step that's needed to push the Internet's security forward: **"Just say NO to port 80"** – which is inherently unencrypted. It got us to where we are today, but for nearly all purposes, its day has passed. We know from everything we've seen that inertia being what it is, nothing ever moves forward on its own. It just doesn't. It's always easier to just leave things as they are. But a more secure future means that organizations such as Cloudflare need to take a leadership stand once it becomes feasible to do so, and it's finally feasible today.

Malware switching to obscure languages

An interesting newly published research paper by researchers out of Greece and the Netherlands caught my attention. Its title is: *"Coding Malware in Fancy Programming Languages for Fun and Profit"* I'm going to share the paper's Abstract and its introduction which will give us a sufficient sense for what these researchers have found. The Abstract explains:

The continuous increase in malware samples, both in sophistication and number, presents many challenges for organizations and analysts, who must cope with thousands of new heterogeneous samples daily. This requires robust methods to quickly determine whether a file is malicious. Due to its speed and efficiency, static analysis is the first line of defense.

I'll interrupt here to mention that broadly, code can either be examined statically, which is just looking at the code bytes themselves after loading them into memory but NOT actually running the code, or, dynamically, which entails creating a sandbox of some sort, often an industrial-strength virtual machine, to actually run the code after it has been loaded to examine the code's actual behavior when it is run. Not surprisingly, static analysis is much faster and efficient when it can be done. The Abstract continues:

In this work, we illustrate how the practical state-of-the-art methods used by antivirus solutions may fail to detect evident malware traces. The reason is that they highly depend on very strict signatures where minor deviations prevent them from detecting shellcodes that otherwise would immediately be flagged as malicious. Thus, our findings illustrate that malware authors may drastically decrease the detections by converting the code base to less-used programming languages. To this end, we study the features that such programming languages introduce in executables and the practical issues that arise for practitioners to detect malicious activity.

The paper's introduction provides some more interesting background:

In the past decade, malware has undergone significant changes. The main drivers of these changes can be attributed to the vast digitization of products and services and the development of a payment system that allows anonymous transactions to bypass the protections of the traditional banking system. [In other words, CryptoCurrency] *This has boosted the number of possible victims and the potential impact of malware. Moreover, anonymous payment methods enable a wide array of illicit transactions to be performed, which, in the case of malware, is the apparent case of ransomware.*

Both the US Cybersecurity and Infrastructure Security Agency (CISA) and the European Union Agency for Cybersecurity (ENISA) have recognized malware as the top cyber threat. Indeed, malware attacks impact our everyday lives by harvesting sensitive information, crippling critical services, and causing significant damage to individuals and corporations. This has placed malware in a pivotal role in the crime ecosystem and created an individual ecosystem with independent roles operating in a business model called Malware-as-a-Service.

*The security industry's response to the abovementioned threats is collecting and analyzing malware samples. At a rate of around **280,000 malware samples per day in 2024**, which is more or less similar to previous years, static analysis remains the most effective and profound remedy to detect malicious files quickly. In this arms race between malicious actors and defenders, the development of malware has evolved into an underground industry to bypass security controls by employing malware authors and monetizing the infected hosts.*

Of course, bypassing static analysis does not grant them a foothold to the targeted host. Nevertheless, it significantly raises their chances of achieving their goal, as they then often need to bypass behavioral checks. Although endpoint detection and response systems usually apply such checks, and vendors often portray them as silver bullets, there are several ways to bypass them. In this work, we limit our scope to static analysis.

Even though malware written in C continues to be the most prevalent, malware operators, primarily well-known threat groups such as APT29, increasingly include non-typical malware programming languages in their arsenal. For instance, APT29 recently used Python in their Masepie malware against Ukraine, while in their Zebrocy malware, they used a mixture of Delphi, Python, C#, and Go. Likewise, Akira ransomware shifted from C++ to Rust, BlackByte ransomware shifted from C# to Go, and Hive was ported to Rust. According to reports, the result of these changes was exhibited increased resistance to reverse engineering and a reduced detection rate or misclassification.

On other occasions, C-language malware families are not recreated from scratch. Instead, malware authors write loaders, droppers, and wrappers in "exotic" languages. This provides them with several advantages such as bypassing signature-based detection, so they can effectively wrap their payloads within harder-to-detect shells that are newly built. Thus, attackers continue to use the same initial penetration vector and a significant portion of their methods, suggesting that threat actors prefer to transfer the original malware code to different languages instead of modifying their tactics, techniques, and procedures (TTPs) to avoid detection.

This approach allows them to maintain the effectiveness of their attacks while remaining under the radar of security systems. Since these languages may be less widely recognized or understood, they add an extra layer of obfuscation to malware, making it harder to detect and analyze. Furthermore, security analysts have reported increased difficulty in reverse engineering such malware samples due to reprogramming efforts. Thus, combining different languages and obfuscation techniques complicates dissecting and reverse engineering the malware's structure, functionality, and intent.

Our work explores the problem of detecting malware written in uncommon languages using a data-driven approach. Rather than merely reporting and examining this trend, we performed a targeted experiment by writing malicious samples in different programming languages and compilers and drilling down to the distinctive characteristics. This analysis practically shows the unique features that adversaries gain and highlights the emerging issues for malware detection and analysis.

This work led to the formulation of some interesting research questions that have not been systematically studied in the academic literature, and we try to answer them in this work:

- *Research Question 1: How does the programming language and compiler choice impact the malware detection rate?*

- *Research Question 2: What is the root cause of this disparity?*

- *Research Question 3: Are there any other benefits to an attacker shifting the codebase to less common programming languages and compilers beyond the detection rate by static analysis?*

What they learned was quite interesting. They created their own malware using the top two

malware exploit techniques that have been identified across the industry and they implemented the underlying malware concept in every language imaginable – even LISP, Leo! Here's what their extensive research concluded, and the answers they arrived at for each of their three research questions. They wrote:

> *Malware is predominantly written in C/C++ and is compiled with Microsoft's compiler. However, answering RQ1 with our experiments — "How does the programming language and compiler choice impact the malware detection rate?" — our work practically shows that by shifting the codebase to another, less used programming language or compiler, malware authors can **significantly decrease** the detection rate of their binaries while simultaneously increasing the reverse engineering effort of the malware analysts. It is crucial to note that the malware authors do not necessarily need to radically change their codebase, as, for instance, just the choice of using a different compiler, even for famous programming languages like C, can have the same impact. Our experimental results illustrate that there are significant deviations in how programming languages and compilers generate binaries, and that they can serve as an additional layer of obfuscation for malware authors.*

In other words, since nearly all of the malware code is written in C and compiled using Visual Studio (so it said in their paper), the static analysis AV detectors have all been similarly oriented for that assumption. So simply by switching to, Turbo C, GCC or WATCOM C, those assumptions about the binary code produced will be broken and AV detection rates will drop without any need to rewrite the malware. (I'm not sure it was good for this paper to be published, but it was true either way, and as they said at the top, they're already seeing malware moving to other languages.) Their Research Question 2 asked about the root cause of the disparity. They wrote:

> *The root cause for the disparities that we raise (RQ2), as highlighted with our use case in Haskell and the metrics for each tested pair of programming language and compiler, is that there are radically different ways that each of them reaches the same result. For instance, different ways of storing strings and different approaches in the internal representation of functions can render many static detection rules useless. As a result, there is no "one-size-fits-all" approach, so further research is necessary to systematically identify these differences and group them.*

Okay. So essentially their saying that since static code analysis is constrained to simply examine code that's lying there in RAM, things such as function calling methods which pass parameters in different ways, or static strings that are stored and represented in differing ways, all of which will vary by language, will all serve to dramatically confuse status analysis. It might result in false positive detections. But it's just as likely to allow bad code to slip past.

Answering their final research question: *"Are there any other benefits to an attacker shifting the codebase to less common programming languages and compilers beyond the detection rate by static analysis?"* They write:

> *Answering question 3, this shift in languages may come with additional benefits for attackers. An obvious case is cross-compilation and multi-platform targeting languages, which enable malware authors to build a single malware variant and have it compiled for multiple operating systems. This strategy can significantly reduce the time and number of tools needed to achieve their objectives, thereby expanding the scope of any hostile campaign. IoT devices, in particular, support a range of CPU environments, making it necessary for malware targeting these devices to be compatible with not only x86 and x64 architectures but also various other*

architectures such as ARM, MIPS, m68k, SPARC, and SH4.

A typical example is Mirai, which uses GCC, yet one of its successors, NoaBot, uses uClibc-based cross-compiler and is statically built to target embedded Linux systems. In this regard, other options could be more efficient. For instance, Go can be cross-compiled to all major operating systems, as well as Android, JavaScript, and WebAssembly. One of its advantages is that it provides statically compiled binaries by default, eliminating runtime dependencies and simplifying deployment on target systems. Go also features a robust package ecosystem that allows developers to easily pull in code from other sources. In general, cross-compilation in Go is as simple as setting two environment variables, making it almost trivial to modify the build process to produce binaries for every major platform. As a result, malware can be developed at a faster rate, targeting a broader range of architectures and systems. Indeed, HinataBot, another descendant of Mirai, is developed in Go to take advantage of the above. HinataBot's discovery was much more difficult as a result. Unfortunately, the bar to creating a new variant of Mirai using Go or other languages is now quite low. This allows criminal groups to create their own variations.

Beyond cross-compilation, there are several other reasons to witness more changes in the malware codebase. After all, malware developers, like any other software engineers, have specific needs when choosing programming languages and tools. Different languages offer various benefits for different scenarios, and the choice of language can significantly impact the development and functionality of malware. For instance, built-in security mechanisms and type safety may be prioritized by ransomware authors who want to avoid leaks of the encryption keys to guarantee that their victims will not be able to develop decryptors. A typical example is Rust, which offers built-in memory mechanisms to prevent common vulnerabilities and type safety.

Oh my god. So even malware is now benefiting from the enhanced memory management and security created through the use of more modern and safe languages! Wonderful. They wrote:

Other aspects can include library availability; facilitating interaction with the underlying operating system and enabling critical malware functions, low-level access, and control over memory layout; having full control over the malware's behavior and performance but also direct compilation to machine code; creating an executable file directly and use other tools for obfuscation.

While shifting to another programming language may seem complicated, especially when considering less popular ones, large language models (LLMs) ...

(Oh, boy ... in other words, AI)

... may come to the rescue; after all, they have proven their capability for generating code quite accurately and various cybersecurity tasks, and malicious actors are abusing them. As a result, they can translate code from one programming language to another, requiring little fine-tuning. This way, malware authors can seamlessly develop loaders, droppers, and other components in languages they may not be familiar with.

It is true that the malware that we examine in this work represents a small fragment of the total; nevertheless, it is stealthier and introduces more bottlenecks for the reverse engineer. Given that the APT groups are shifting their codebases and the malware-as-a-service model

> *facilitates the trading of malware so different malware mixtures per campaign can be purchased, this diversification is expected to continue. By disregarding these samples and only focusing on traditional programming languages and compilers, we provide malware authors with an effective hideout that they can easily exploit. Therefore, we believe that a deeper analysis of the executables produced by other compilers and programming languages is needed to improve detection rates but also develop better reverse engineering tools.*

So that's all very interesting. I wouldn't say that they discovered anything earth-shattering or surprising. Their results are pretty much what we would expect. But some of the tricks they highlighted, such as simply recompiling unchanged malware under a different compiler for the same language was interesting. And by clearly demonstrating **in fact** what we might assume, their work should serve to get the authors of static AV detection to consider whether they will be needing to broaden the scope of their detectors. It's also clear that as malware moves away from the traditional Microsoft Visual Studio 'C' to other less commonly used languages, the static detection rates for any AV that doesn't keep up will be dropping.

In reading this, the one language that I didn't see, which would have been really interesting, would have been FORTH. Based upon what these researchers found, I would imagine that FORTH would have a number of advantages for malware. For one thing, FORTH only needs a very small and readily available runtime interpreter that's has already been ported everywhere.

I often refer to FORTH as a "write only" language because it's impossible to read it. It's a super-dense stack-based interpreted language and even the people who write FORTH code once are later unable to understand what they wrote when they read it later. I've never encountered a more hostile language. So I would bet that static code analysis would be likely to completely fail for FORTH. And FORTH is entirely platform & machine independent. The one problem for the bad guys with writing malware in FORTH is that you really do need to know what you're doing. FORTH is not difficult to use, but neither does it hold your hand ... in fact it slaps it away.

People are still reusing passwords
Cloudflare recently published a piece of research that I wanted to share. I was initially confused by the headline of their blog post, which read: *"Password reuse is rampant: nearly half of observed user logins are compromised."* And I thought, what do you mean "*nearly half of observed user logins are compromised*"?? It turned out that the problem with their headline was the somewhat unclear word "compromised." A better choice may have been to say *"nearly half of user logins use previously leaked passwords."* In other words, passwords that are likely known by Troy Hunt's Have I Been Pwned site. Cloudflare wrote:

> *Accessing private content online, whether it's checking email or streaming your favorite show, almost always starts with a "login" step. Beneath this everyday task lies a widespread human mistake we still have not resolved: password reuse. Many users recycle passwords across multiple services, creating a ripple effect of risk when their credentials are leaked.*
>
> *Based on Cloudflare's observed traffic between September - November 2024, 41% of successful logins across websites protected by Cloudflare involve compromised passwords. In this post, we'll explore the widespread impact of password reuse, focusing on how it affects popular Content Management Systems (CMS), the behavior of bots versus humans in login attempts, and how attackers exploit stolen credentials to take over accounts at scale.*

I'm going to skip over most of this because everyone listening to this podcast already well-

understands the dangers of password reuse and I'm sure that everyone listening is now using some form of password manager which is able to synthesize complete gibberish passwords on the fly for use, then store and later reuse.

One thing I wasn't appreciating before this was the size to which Cloudflare has quietly grown. At one point they wrote:

> *Our data analysis focuses on traffic from Internet properties on Cloudflare's free plan, which includes **leaked credentials detection** as a built-in feature. Leaked credentials refer to usernames and passwords exposed in known data breaches or credential dumps — for this analysis, our focus is specifically on leaked passwords. With 30 million Internet properties, comprising some 20% of the web, behind Cloudflare, this analysis provides significant insights.*

Wait. 30 million Internet properties, one out of every five, are now running their traffic through Cloudflare? Wow. That crept up on me. So then they explain:

> *One of the biggest challenges in authentication is distinguishing between legitimate human users and malicious actors. To understand human behavior, we focus on successful login attempts (those returning an HTTP 200 OK status code), as this provides the clearest indication of user activity and real account risk. Our data reveals that approximately 41% of successful human authentication attempts involve leaked credentials.*
>
> *Despite growing awareness about online security, a significant portion of users continue to reuse passwords across multiple accounts. And according to a recent study by Forbes, users will, on average, reuse their password across four different accounts. Even after major breaches, many individuals don't change their compromised passwords, or still use variations of them across different services. For these users, it's not a matter of "if" attackers will attempt to use their compromised passwords, it's a matter of "when".*

And they note that, as we would expect, automation, in the form of 'bots, are the primary abusers of leaked credentials, writing:

> *Bots are the driving force behind credential-stuffing attacks, the data indicates that 95% of login attempts involving leaked passwords are coming from bots, indicating that they are part of credential stuffing attacks. Equipped with credentials stolen from breaches, bots systematically target websites at scale, testing thousands of login combinations in seconds.*
>
> *Data from the Cloudflare network exposes this trend, showing that bot-driven attacks remain alarmingly high over time. Popular platforms like WordPress, Joomla, and Drupal are frequent targets, due to their widespread use and exploitable vulnerabilities.*
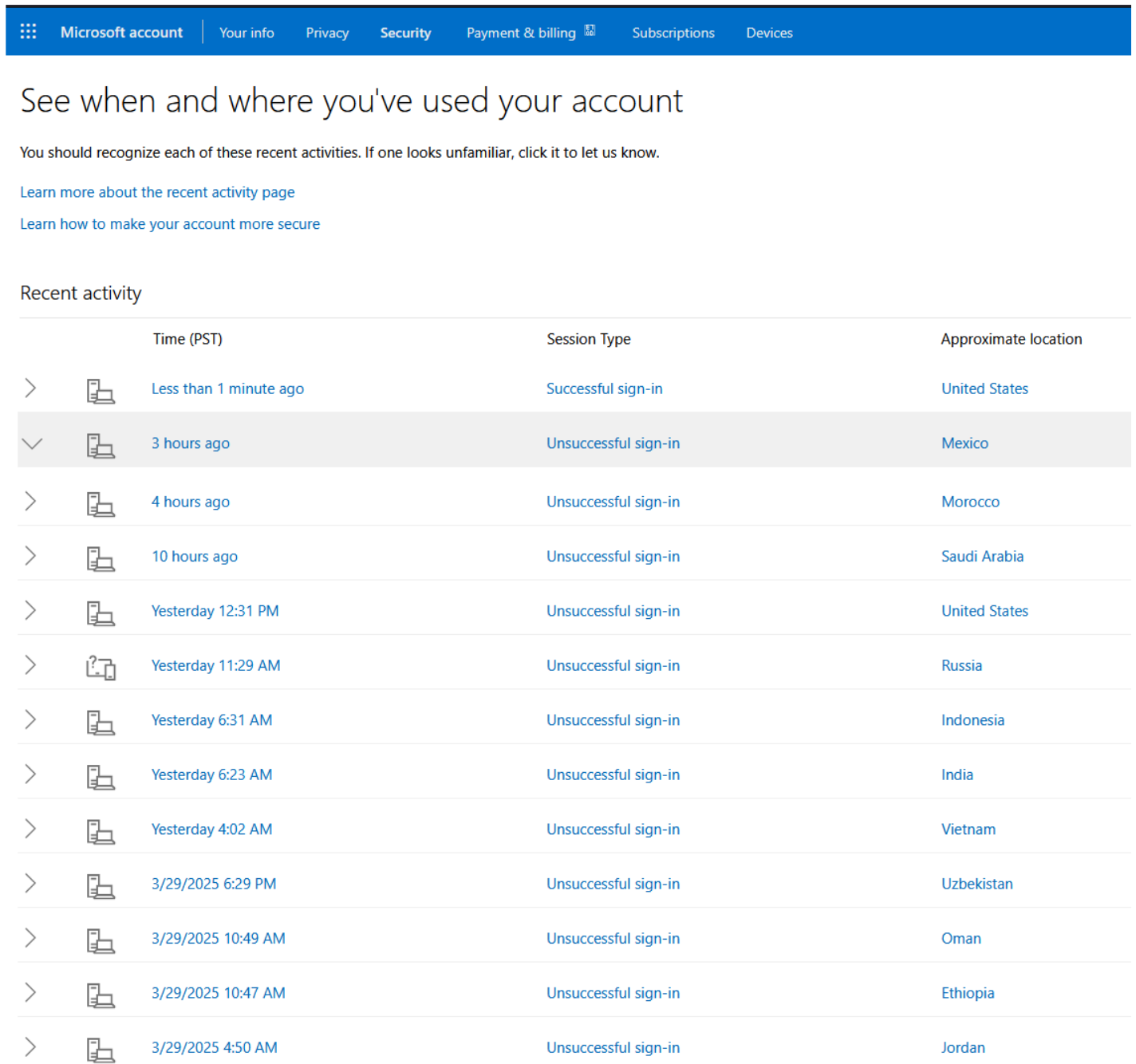>
> *Once bots successfully breach one account, attackers reuse the same credentials across other services to amplify their reach. They even sometimes try to evade detection by using sophisticated evasion tactics, such as spreading login attempts across different source IP addresses or mimicking human behavior, attempting to blend into legitimate traffic.*
>
> *The result is a constant, automated threat vector that challenges traditional security measures and exploits the weakest link: password reuse.*

By coincidence, one of our listeners, Jeremiah Albrant, sent a piece of feedback yesterday

morning with the subject: *"Microsoft / Hotmail account password stuffing attempts are very real"*. In his email, he wrote:

Talking to some co-workers they showed a screenshot of their sign-in activity from their Microsoft account so I checked mine. I was blown away. My own screenshot is below. The successful attempt is my own. Clicking through each unsuccessful attempt shows they entered the wrong password. I am so glad I use unique passwords for my accounts. This is nuts.

	Time (PST)	Session Type	Approximate location
	Less than 1 minute ago	Successful sign-in	United States
	3 hours ago	Unsuccessful sign-in	Mexico
	4 hours ago	Unsuccessful sign-in	Morocco
	10 hours ago	Unsuccessful sign-in	Saudi Arabia
	Yesterday 12:31 PM	Unsuccessful sign-in	United States
	Yesterday 11:29 AM	Unsuccessful sign-in	Russia
	Yesterday 6:31 AM	Unsuccessful sign-in	Indonesia
	Yesterday 6:23 AM	Unsuccessful sign-in	India
	Yesterday 4:02 AM	Unsuccessful sign-in	Vietnam
	3/29/2025 6:29 PM	Unsuccessful sign-in	Uzbekistan
	3/29/2025 10:49 AM	Unsuccessful sign-in	Oman
	3/29/2025 10:47 AM	Unsuccessful sign-in	Ethiopia
	3/29/2025 4:50 AM	Unsuccessful sign-in	Jordan

If others want to see their history, I clicked on my avatar in the top right from my inbox, then "My Profile" then the "Security" tab then "View my sign-in activity". Unfortunately the UI is primitive and doesn't seem to have filter or sorting options so unless I click the view more activity link over and over while expanding each item I don't see any other way to determine whether somebody has my password and just failed to get past the 2FA. In other words, it's necessary to expand each attempt to determine the cause of the login failure.

Jeremiah's login log shows about five attempts per day. At the top we see his successful login showing its location in the United States. Three hours before that was a failed attempt made from an IP address in Mexico. An hour before that, Morocco. Six hours before that, Saudi Arabia. The previous day attempts were made from the US, Russia, Indonesia, India and Vietnam. And the day before that we see Uzbekistan, Oman, Ethiopia and Jordan.

Given that, the most obvious security feature for Microsoft to implement would be account access geofencing. But my quick search revealed that not only is there massive demand for this from everyone and anyone who examines the history of failed login attempts against their accounts – as our listener Jeremiah did – but that geofencing is only available for business class accounts and not to individual users. That's difficult to explain since anyone examining the history of failed authentication attempts should be infuriated by their inability to block all such obviously bogus authentication attempts from across the globe.

As I said, I have no doubt that, just like Jeremish, all of our listeners are using unique gibberish passwords with the help of a password manager. With a heaping helping of multi-factor authentication piled on top. But we all know that most of our friends and family are not listening to this podcast. So this amounts to a gentle nudge reminder for us to proactively annoy all of them with this. It really would be to their benefit. And shame on Microsoft for not allowing end-users to lock their account authentication to their local country of origin. That's just nuts and so irresponsible.

23andMe follow-up
Following up on last week's mention of 23andMe, I ran across a bit more information in a security newsletter. Under the headline *"23andMe files for bankruptcy after mega-hack"* it said:

> *DNA and genetic testing service 23andMe has filed for bankruptcy 15 months after experiencing a major data breach. The company has been losing money for years, but its problems were amplified last year after a series of class-action lawsuits related to the breach. Its entire board resigned last year, its CEO last week, and the company is now attempting to sell itself under the supervision of a court. The company has DNA profiles on over 15 million users. Privacy regulators across the US and Europe are now urging users to request the deletion of their data before it's sold.*

So I'll just remind our listeners that once you login to your 23andMe account, you can use the GRC shortcut I created: grc.sc/byebye to immediately jump to the page containing the various account data dumping and deleting options. Not a house on fire, but why not do it if they have your data, and if some unknown entity will be purchasing it in the future?

https://www.ransomlook.io/recent
I was pursuing information about a new-on-the-scene ransomware group calling itself Arkana. Arkana's first victim was WOW, one of the largest ISPs in the US. But in following some trails I ran across a site I hadn't seen before and which we've never talked about before. It's ransomlook.io so https://www.ransomlook.io and because what you'll find there is both chilling and somewhat astonishing, I made it this week's shortcut of the week: grc.sc/1019. The site has been around since 2022, they're on Mastodon and BlueSky and a huge amount of work has been put into the site. Once you get to the homepage, under "Groups Profiles" you'll find listed there, every group we've ever talked about and hundreds more lesser groups or newer group that we haven't. There are many familiar names. The "Ransomware Notes" section lists all of the various

notes that the various ransomware groups have sent to their victims. And I mentioned "chilling"? Most chilling of all is the "Recent Posts" page which contains a listing in reverse chronological order, starting with the most recent, of the latest ransomware victims and which group took them down. As I'm writing this, yesterday, at 3pm on Monday, March 31st, there are 22 new ransomware victim companies listed, by name, just for today up to this point and I don't know what time zone they're using. But listed there in black and white are the corporate names and domain names of many victims.

There's no way to come away from a perusal of this site without the very clear knowledge that the ransomware category of criminal cybercrime is very much a going concern. And it's also sobering to appreciate that not one of those companies wanted to have their security breached and penetrated and their internal proprietary data stolen, yet it was nevertheless.

Listener Feedback

A reminder about "InControl"

Hi Steve! Just thought I'd send you a quick message to let you know how thankful I am for your incredibly useful little program InControl!

I am an avid flight simulator enthusiast, and the best way to enjoy flight simulation these days is with a high end VR Headset. As such I have a HP Reverb G2 V2 headset, which when new in 2021 was several hundred £ or $. This headset uses Microsoft's "Windows Mixed Reality" platform which is built into windows. While the headset itself is excellent the WMR platform was somewhat of a failure for Microsoft, With most other manufacturers using other platforms. Despite that MANY people in the flight sim world still use the Reverb G2 with WMR because of its high resolution.

In their infinite wisdom Microsoft have decided to remove WMR from Windows 11 from update 24H2. Rendering all WMR Headsets like my HP Reverb completely useless. Indeed friends of mine have the update only to find their VR headsets no longer work and they have to go through the huge hassle of somehow stepping back to 23H2 to get their setups working again. Thankfully with InControl we can stay on 23H2 and retain the WMR functionality!

I've recommended InControl to several of my friends and it seems to do the trick of MS forcing them to update against their will. Sorry for the long e:mail. But many thanks for your work! Cheers, Ben Dean, UK

EU OS

Robert Riemann is the Head of Sector for Digital Transformation in the Technology and Privacy Unit at the European Data Protection Supervisor in Brussels. He contributes to the overall IT governance of the EDPS (European Data Protection Supervisor) and supports the EDPS representation in several EDPB subgroups. His CV indicates that he holds a PhD in computer science with a thesis on distributed protocols for aggregation of confidential data with applications in, for example, online voting. And he has his Masters in Physics from Berlin's Humboldt University.

As the title of this podcast suggests, Robert is spearheading a well thought out departure from EU dependence upon Microsoft Windows. The site where this is being organized calls itself the European Union's home for their free public sector personal computing operating system, highlighting three key features for this project: Secure, Sovereign and Sleek.

- Secure – means an OS built from open source that does not phone home.
- Sovereign – means an OS built to the requirements for the EU public sector.
- And Sleek – means an OS that is fast and eco-friendly on new **and** old hardware.

Obviously, none of those goals are met by Windows. On that home page they ask the reader the question: What is EU OS? Their answer is:

> *EU OS is a Proof-of-Concept for the deployment of a Fedora-based Linux operating system with a KDE Plasma desktop environment in a typical public sector organisation. Other organisations with similar requirements or less strict requirements may also learn from this Proof-of- Concept. Despite the name, EU OS is technically not a new operating system. Distrowatch lists currently over 250 Linux operating systems ('distributions'), not counting their many various flavours, spins or subvariants. The added value of EU OS is a different one:*
>
> - *a common Linux OS as a base for all EU OS users with options to layer on top modifications (national layer, regional or sector-specific layer, organsation-specific layer)*
> - *a common desktop environment*
> - *a common method to manage*
>   - *users and their data*
>   - *software*
>   - *devices*

The site at https://eu-os.gitlab.io endeavors to full articulate the goals of this initiative:

> *When at the beginning, the user base is still too small to pool enough resources to take care of the EU OS (base version) within the public sector, it may be possible to contract commercial support for maintenance. For this reason, the EU OS Proof-of-Concept proposes to choose an upstream Linux OS with options for commercial support.*
>
> *EU OS is not the first to propose a Linux-based operating system for the public sector. The motivation is often the same and can be looked up from projects like GendBuntu and LiMux.*
>
> - *'public money – public code' means the public investment profits the entire public (and private) sector*

- *synergy effects lead to tax savings, because there is no per-seat license cost*
- *independence from software suppliers and vendor lock-in*
- *independence in scheduling software migrations and potential hardware upgrades*
- *deploy new technologies with controlled cost*
- *use of open standards to foster innovation*
- *better use of IT administration resources (reportedly for the French use case with 90,000 seats)*
- *ability to do own code analysis*
- *worldwide free software community*

The project lists its philosophical goals as:

- *use of open source*
- *use of desktop environment KDE Plasma (though Gnome as alternative not excluded)*
- *use of Gitlab*

They're leaving the entire scope of the project somewhat open-ended writing: *"There is no clear scope yet and the scope may evolve in the future. But the rule of thumb so far: In scope is everything necessary to deploy a Linux-based operating system to an average public body with a few hundreds of users."* And they do give examples of what is clearly out of scope:

- *The development of a novel Linux operating system ('distribution') from scratch. Instead, EU OS should build on top of an existing well established Linux distribution.*

- *The deployment of EU OS outside of a corporate environment. For their personal computers, people can already choose between a large variety of Linux distributions.*

- *The deployment of EU OS on other devices than typical desktop workstations or laptops. Hence, smartphones are out of scope.*

Looking at use cases and at some previous attempts and successes, the site notes "To make EU OS a success, it should support a large number of use cases and consequently a large user base. This helps to gather political support and funding for continuous improvements and innovation."

They note that some regions outside of Europe have already realized the benefits of an operating system under their control:

- **Astra Linux** is a Russian Linux-based computer operating system (OS) that is being widely deployed in the Russian Federation to replace Microsoft Windows.

- **Kylin** is an operating system developed by academics at the National University of Defense Technology in the People's Republic of China since 2001. Together, Kylin and Neokylin share a 90% market share of the government sector.

- **Nova Linux** was central to the Cuban government's desire to replace Windows. Hector Rodriguez, Director of University of Information Science in Havanna, said that "The free software movement is closer to the ideology of the Cuban people, above all for independence and sovereignty." Other cited reasons to develop the system include the United States embargo against Cuba which made it difficult for Cubans to purchase and update Windows,

as well as potential security issues feared by the Cuban government because of the U.S. government's access to Microsoft's source code.

Citing these use-case successes, the site states: *"This leaves no doubt about the feasibility of large-scale Linux deployments in the public sector. It is only a matter of political support, priority and funding."* The site notes some details of past migrations away from experiences with Microsoft and Windows:

City of Munich
This was 20 years ago, but it serves to highlight the problems inherent in the use of another country's commercial operating system for public sector needs. The report wrote:

The city of Munich is migrating its desktop computers from Windows to GNU/Linux. After preparations began in 2003, the city's basic client, a customized version of Debian GNU/Linux, is being deployed on a growing number of PCs since fall 2006. The LiMux project puts great emphasis on becoming independent from software suppliers. Florian Schießl, the deputy project coordinator for LiMux, explains: Microsoft has shown us what it means to be dependent upon a vendor.

Until 2003, the city was using Microsoft Windows NT 4 across the board, and was by and large satisfied. When Microsoft decided to end the support for this operating system, this meant that hardware and important procedures would eventually stop working. It was from this experience of being totally at the mercy of an external party that we wanted to take the road to more independence.

For the **French Gendarmerie**

Gendbuntu is possibly one of the largest Linux-on-desktop deployments in the EU public sector with about 82,000 seats. Lieutenant Colonel Guimard said: "Moving from Microsoft XP to Vista would not have brought us many advantages, and Microsoft said it would require training of users. Moving from XP to Ubuntu, however, proved very easy. The two biggest differences are the icons and the games. Games are not our priority."

"The transition [to Linux] went unexpectedly smoothly. Almost no additional training was required for the local police forces using the computers in their daily work. The Ubuntu user interface was easy to get used to. Pascal Danek points out that a transition from Microsoft Windows 2000/XP to Vista would have been more difficult, since the new version of that OS introduces many new features and designs which might confuse users."

The French Gendarmie currently uses a customised version of Ubuntu called "Gendbuntu". If EU OS would be used instead, resources could be mutualised across all users of EU OS.

One of the references for this is an Arstechnica piece from 2009 with the headline "French police saves millions of euros by adopting Ubuntu". And it's not difficult to imagine at this point that they're glad they did that back then. They're likely still running on the same hardware.

Then we have the case of the **Swiss Federal Court** 🇨🇭

"Until 2001 the court had a simple all-in-one IT platform, which lacked greatly in functionality and ultimately became outdated. The Court's IT direction thus saw the necessity to introduce a

new IT infrastructure that would ensure sustainable standards in the future. During the analysis done as part of the planning process, open source software emerged as more sustainable than proprietary software, especially with regard to modularity and file formats. The use of open source software also ensured vendor independence and security, which are two very important aspects for a court. In 2001, the new IT system running on the operating system Solaris by Sun Microsystems was introduced. With this also came the introduction of the office suite StarOffice, the internet browser Firefox, and the email client Novell Evolution, besides other more specialised applications. At the early stages of the migration, users had to get used to the new programs, but as the migration from the previous system brought numerous improvements, the process went relatively smoothly and was broadly accepted. Where some doubts about open source software existed in the beginning, these have mostly faded by now.

Linux Plus 1 in **Northern Germany**

A region in the north of Germany is currently preparing the migration of their entire public administration to a Linux desktop. This migration could become one of the largest Linux-on-desktop deployments in the EU public sector with 30,000 of seats. It is unclear which operating system will be used. Rumors say it will be based on KDE Plasma. If EU OS would be used, resources could be mutualised across all users of EU OS.

A reference listed for that was a piece in the ever irreverent Register last April with the headline: *"Germany's Northernmost state ditches windows."*

CERN

The US Fermilab and the European Organization for Nuclear Research, known as CERN, already maintained their own operating system in the past: Scientific Linux. Unfortunately it reached end-of-life last June, 2024. While CERN certainly can manage their devices without the support of EU OS, collaborating with the large hub of innovators would certainly strengthen EU OS to the benefit of all.

On the site they have a box with the link titled: *"Click to reveal more references with migration projects"*. So I clicked it and it opened a surprisingly long list of case histories – many dozens – where the hard decision was made to switch away from Windows to Linux. And browsing through those stories, there's no sense of regret evident.

And, as we know, there is still a lock-in problem with Microsoft's otherwise very compelling solutions. Under the heading *"Cities and Communities"* they note:

Only a few cities have migrated to Linux so far. Compatibility with the federal government and the plethora of business processes a city owns are a challenge. Oftentimes, reliance is strong on Microsoft Office, which historically did not run on Linux. With Microsoft 365 working in the browser, a work-around may be possible. Few examples of cities are known that want to migrate to Linux or have already migrated. In both cases, EU OS could help.

And that contained a reference link to a page titled: *"List of cities who migrated to LibreOffice."* Since this Robert Riemann is the Head of Sector for a the European Data Protection Bureau, he knows something about the benefits that Data Protection Authorities might obtain. The site says:

> *Data protection authorities have good reasons to switch to EU OS:*
>
> *Leading by Example: EU OS is data protection-friendly and gives the organisations as much control about data flows as they want.*
>
> *Strengthening Independence: EU OS offers data protection authorities to investigate without any conflicts of interests all organisations relying on operating systems other than EU OS. They would just need to switch when the majority would have adopted EU OS.*
>
> *Ahead of migrations to Windows 11, data protection authorities wonder about alternatives, no data protection authority has yet announced to migrate to free alternatives. Many data protection authorities rely on central administration for their IT requirements and have little influence over the process.*

And as for Computer Emergency Response Teams (CERTs):

> *Computer emergency response teams are often full of technical experts that usually have some knowledge about Linux already. They may benefit from the flexibility and tools that Linux offers to carry out their investigations and analyses. EU OS could be a good fit.*

And, finally, for European Institutions"

> *A specific goal of EU OS is to get on the desktop of the administration of the European Union, including the European Commission, Parliament and Court of Justice. However, so far no migration plans to Linux are publicly known yet.*

I suspect that such core institutions are more likely to be followers than leaders.

Their FAQ page offers some interesting technical insights:

Is EU OS another Linux distribution that I can try out?

> *EU OS is not another Linux distribution. EU OS is a community-led Proof-of-Concept, which employs existing Linux distributions. The challenge of the proof is not that an individual can use Linux on their own computer. Instead, the challenge is to prove that an admin team can manage users and their data, software and devices with or without Active Directory and without Microsoft Windows within a migration period of 2 years rather than 20 years.*
>
> *For this, EU OS wants to propose a common Linux OS and desktop environment as a base and, more importantly, a common method to manage users and their data, software and devices.*
>
> *EU OS is not meant for home users, but for system administrators who want to automatically deploy and manage Linux across many corporate computers/laptops.*

How can EU OS achieve its goals of being secure and sovereign when it relies on software from other countries (e.g. the US)?

EU OS shall not confound sovereignty and protectionism. There is no problem per se in relying on international free and open source software (FOSS) components, and oftentimes it is in practice unavoidable. However, EU OS promotes the maintenance of strict control over business data and telemetry data. This includes the free choice where to store such data (on-premise or cloud of choice). Furthermore, the availability of know-how for a given FOSS component within the EU shall be considered. It remains to be studied if EU OS FOSS components (such as the Linux kernel, systemd, Wayland, Pipewire, Fedora or AlmaLinux), could face export limitations, which would pose a threat to the sovereignty offered by EU OS. Such threads cannot be mitigated by EU OS alone and should be addressed through industry supply chain security policy.

Why does EU OS propose to rely on Fedora-based Linux distributions?

EU OS is not a product (yet), but only a Proof-of-Concept. The choice of the employed base Linux distribution or desktop environment (Gnome or KDE) is not a core concern as it does not impact much how admins manage users and their data, software and devices. Nevertheless, EU OS cannot avoid picking one base Linux distribution to start with. Advice has been received and considered from individuals in their personal capacity of the following organisations:

- *EU OS community on Gitlab*
- *CERN*
- *European Commission, DG DIGIT*
- *German Centre of Digital Sovereignty, Zendis (known from openDesk)*
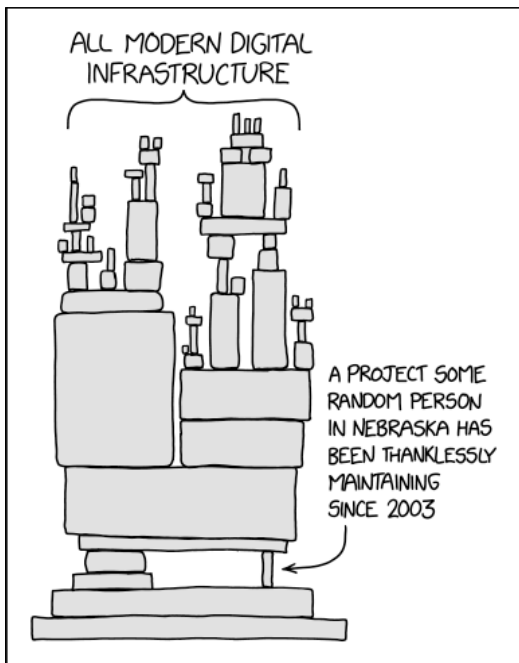- *GnomeOS*
- *openSUSE through their dedicated blog post*

Considering the advice received, the decision was to advance the Proof-Of-Concept with Fedora. For a production deployment after the Proof-of-Concept, any Fedora-based Linux distribution with longer release cycles could be used. Also, a switch to any other bootc-supported Linux distribution would always remain possible.

This effort in the EU is what we would definitely "call handwriting on the wall."

As I was reviewing and assembling all of this, I realized how Windows-centric most of the U.S. is – though I know that many of this podcast's listeners have already liberated themselves from Microsoft's proprietary grasp. But throughout most of the United States, encountering anything other than Windows is a rarity. And that shows little sign of changing. I'm very comfortable with Windows, I love the platform, which I've been using since its birth, and as a commercial product developer it's still where the market is. But I'll also note that I spent time over the weekend updating my Ubuntu system, since I go to whatever lengths are necessary to assure that anything I do also runs smoothly under WINE, the free Windows emulator for Linux.

In the EU, as we saw mentioned, leaving Windows will not be an easy thing for any large organization to do. I suspect that future migration will not be from the top down, but from the bottom up. Smaller entities that are more able to leave Microsoft will be under increasing pressure to do so as Microsoft's nearsighted policies force wholesale hardware replacement when they force software upgrades. This will cause smaller and inherently more flexible entities to explore what alternatives to Windows 11 may exist and having the EU OS present, may provide

a path for smaller organizations to take once they've been pushed.

ALL MODERN DIGITAL INFRASTRUCTURE

A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Okay, now, all that said, I'm haunted by that brilliant and poignant XKCD cartoon which we've looked at from time to time. It's that one showing the massive stack of various sized blocks stacked on top of one another – which is brilliant because it is actually exactly the way modern software "stacks" are created and operate – and where, amid this towering collection, there's one little block off to the side and near the bottom, upon which all of the other blocks implicitly rest.

Wanting to find that XKCD cartoon, I turned to ChatGPT to let it do the legwork for me. I copied and pasted that description above since I'd already written it, and I want to share its reply:

The xkcd comic you're referring to is titled "Dependency" (comic number 2347). This illustration depicts a precarious tower composed of numerous blocks, symbolizing the modern software infrastructure. At the base of this towering structure is a single, small block labeled "A project some random person in Nebraska has been thanklessly maintaining since 2003," highlighting the fragility and reliance of complex systems on often-overlooked components.

The comic serves as a poignant commentary on how critical pieces of modern digital infrastructure can depend heavily on small, open-source projects maintained by individuals without widespread recognition or support. This theme resonates with real-world scenarios where the failure or abandonment of such a project can have widespread repercussions across dependent systems.

I'm 100% certain that everyone listening to this, who has been following along with us for even a few years, will perfectly understand the motivations surrounding the desire to switch **away** from an operating system solution – regardless of how functional, compatible and inter-operable it may be – that does not appear to be directly driven by a motivation of planned obsolescence.

It's one thing to be a computing enthusiast where we're using and working with computers for their own sake, as many, if not most of us are and do. But it's entirely different to be a police station in a small rural town in France where all you want is to be able to bring up records, search the Internet, balance the books and communicate with colleagues. This is a place where a computer is a tool, not a toy, and its reduced ability to be used for "playing games" may be a feature and not a bug.

So it's clear why a move from Windows to Linux would make so much sense. If it's possible for Linux and the tools that run on top of it to get the job done, then it's going to be far more cost effective in the long run, able to keep running effectively and efficiently until the day the hardware finally dies.

But this brings me back to XKCD's observation, that random person in Nebraska, and all of the tens of thousands of other random people who thanklessly create and maintain that system only for the sheer joy of doing so.

I suppose this is a sustainable model. It has always been, after all, the goal of the free and open source software world. That dream is really coming true in spades. But as more and more incredible value is obtained from the tireless work of volunteers, something feels – I don't know, I suppose a bit unfair to them – because to use XKCD's word for it, it really is thankless work.

I've created a great deal of free software which has been and remains quite popular. But it doesn't feel at all thankless to me, because everyone who downloads it knows where it came from and who created it. And I receive sufficient feedback literally in the form of thanks from its users when it surprises them by pointing out an open port on their router, helps them spot a bogus thumb drive, prevents Windows from updating, or lets them know that switching to different DNS servers would speed up their use of the Internet. I receive plenty of thanks.

But I worry about those thankless people who toil without any recognition. I suppose the recognition they receive from their peers within the community they share, is enough. I hope it's enough because having achieved the dreams of the likes of Richard Stallman and Linus Torvalds, what we need now is sustainability. As these thankless developers see more and more of the world using their stuff and taking it totally and literally for granted, I hope they see it as a badge of honor that what they have created is helping so many for such a low cost.

What has been accomplished, as evidenced by the creation of this EU OS unification project is truly a stunning achievement. But now it needs to keep going.