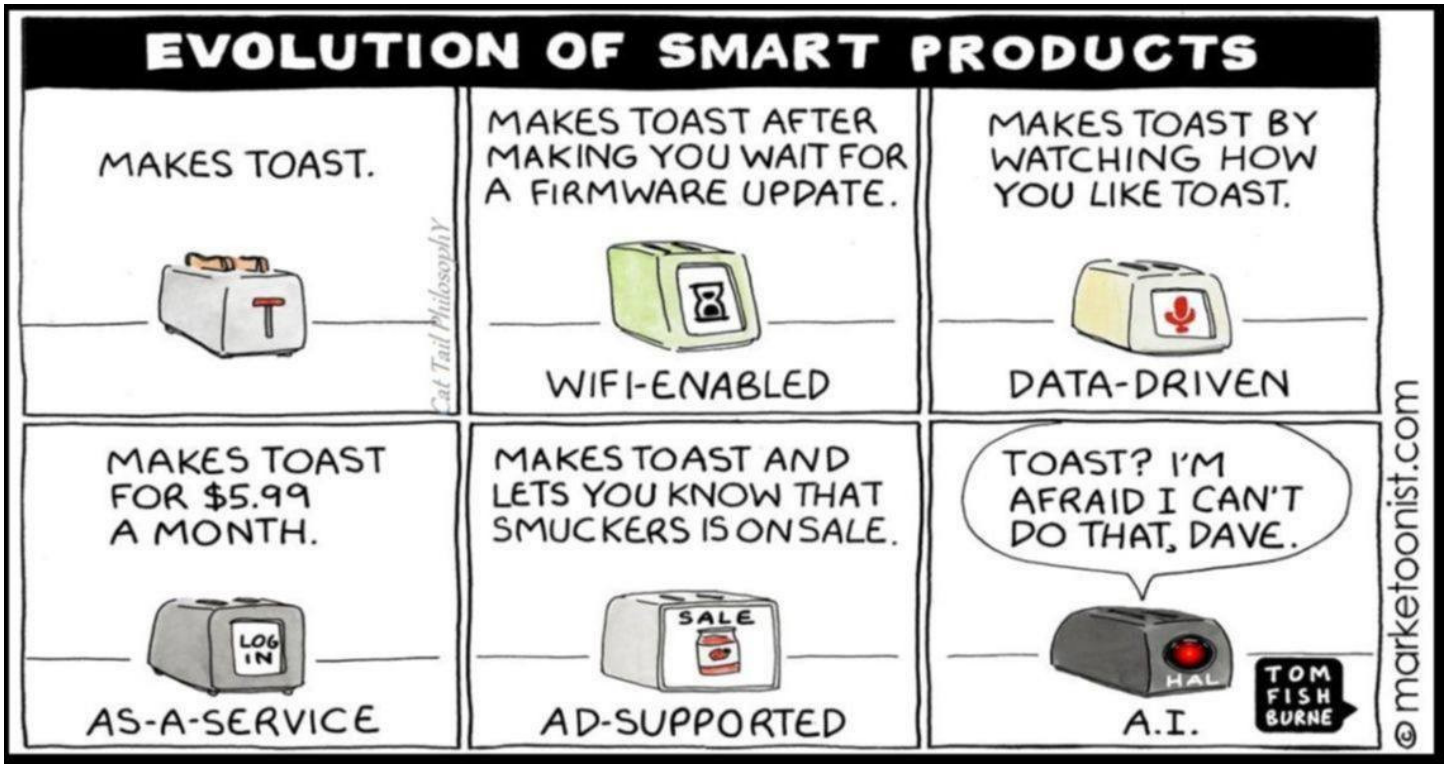
Security Now! #941 - 09-26-23

We told you so!

This week on Security Now!

This week we're chock full of questions! Why is my new ValiDrive freeware not published yet? Why did Apple quietly remove PDF rendering from the Mac after 39 years? Has the NSA been hacking China? What mistake did Microsoft recently make that would require the use of a bigger hard drive? Why did Signal just announce their use of post-quantum crypto? What's the big hurry? Is it possible to create a new web browser from scratch? And if not, why not? Does public key crypto really go both ways? Can pure math generate pure random numbers? One of our listeners believes he has. Could encrypting an entire hard drive then throwing away the key be used in place of the random noise wiping I'm a big fan of? Why hasn't the Unix time problem been fixed yet? Or has it? Will all of the stolen LastPass vaults eventually be decrypted? Am I really leaving Twitter? And, finally... why in the world is this episode titled "We Told You So!" ? The answers to those questions and more will be revealed by the time we're done here today. Welcome to episode #941 of TWiT's Security Now! podcast.



ValiDrive Update

I've received messages from listeners who are anxious to start testing their various USB flash drives with GRC's forthcoming ValiDrive freeware utility. As is usual for me, this side project is taking longer than I expected. And since I paused the completion of SpinRite v6.1, right on the verge of getting it finished, I feel a great deal of pressure to get ValiDrive finished and back to finishing SpinRite. But I don't believe I'm going to look back and feel this was a mistake. All of my previous low-level drive work has been in DOS where it's possible to own the entire machine. It turned out that Windows' USB chain fought back and interfered with ValiDrive's operation much more than I expected. So getting this done correctly was quite involved. But it has made ValiDrive unique because it now incorporates a bunch of technology that's not available elsewhere. And, along the way I've developed a lot of new code that's going to be very useful for future USB work under Windows – like for GRC's secure drive wiping utility. Although that doesn't help us today, it will in the future, and I always take the long view since I plan to be around and active for quite a while. ValiDrive finally appears to be working well and it's being heavily used and tested by GRC's testing group. It has opened my eyes about just how severe this fake USB flash drive problem has become while none of us were paying attention. I'm not quite ready to turn it loose because once it's finished I don't want the distraction of needing to come back to fix little issues that I ignored out of a rush to publish. I want to get back to SpinRite v6.1 and then immediately on to 7. So, things like not saving its reports properly on screens with Windows font sizing set to other than 100%, or some UI text not appearing when a user's screen is set to high-contrast mode, or when a user may be unable to discern similar colors clearly. They're details that I need to put behind me so that what's published will be finished and useful for many years to come. And I mentioned creating and saving reports. I confess that there's also been some feature creep along the way. It's become a very nice and useful utility which I'll formally announce and publish as soon as possible.

So... let's explore what's been happening in the world of security and privacy!

Security News

A Postscript for Postscript in macOS

The Eclectic Light Company blog posted an interesting piece yesterday which they titled *"Postscript's sudden death in Sonoma"* – where "Sonoma" refers to the major macOS release. It had some interesting observations about Postscript as a dangerous interpreter. So I wanted to share what they wrote and I'll comment on the other side:

> *If there's one language that's been at the heart of the Macintosh for the last 39 years it's PostScript, the page description language developed by the founders of Adobe, the late John Warnock and Adobe's team of engineers. It brought the Mac's first commercial success in desktop publishing, in PostScript fonts, and early PostScript printers including Apple's game-changing LaserWriter. Although Mac OS X never inherited NeXTStep's Display PostScript, its descendant Quartz and Core Graphics are still based on PostScript's relative PDF.*
>
> *Following a short illness that started in macOS Monterey 12.3, PostScript has died suddenly in Sonoma. The first sign passed almost unnoticed in Apple's release notes to macOS 12.3, where it recorded the "deprecation" of PostScript in WebKit: "Support for inline viewing of PostScript*

files is no longer available."

Then in macOS 13.0, Preview lost the ability to convert PostScript and EPS: "The Preview app included with your Mac supports PostScript (.ps) and Encapsulated PostScript (.eps) files in macOS Monterey or earlier. Starting with macOS Ventura, Preview no longer supports these files. Other apps that can view or convert .ps and .eps files are available from the App Store and elsewhere." Finally, the complete removal of support for PostScript and EPS was recorded as another "deprecation" in the release notes for Sonoma.

PostScript is an old stack-based interpreted language designed at a time when code security had barely been conceived, and malicious software hardly existed. Among its attractive features is the fact that any PostScript object can be treated as data, or executed as part of a program, and can itself generate new objects that can in turn be executed.

I'll just pause here to observe that Windows Metafiles, which turn Windows drawing primitives into an interpreted format, originally had the same capability of executing code. That's just the way things were done back then. When this was rediscovered much later, everyone freaked out, thinking it was a horrific bug and many people thought I was nuts when I calmly observed that it was clearly, originally, deliberate. Definitely a bad idea today; but entirely reasonable at the time. Everyone just forgot it was there. As this article says: *"PostScript is an old stack-based interpreted language designed at a time when code security had barely been conceived, and malicious software hardly existed."* Exactly.

One other note: Although stack-based languages can be brittle, back at the time, defining a stack-based language was a terrific choice because stack-based representations and interpretations can be incredibly dense and efficient. That's what you would want in a page description language where there are effectively no processor cycles and no easy memory, PostScript's design was brilliant.

Their article continues:

More recently, security researchers have drawn attention to the fact that Postscript is a gift for anyone wishing to write and distribute malicious code. As it's effectively an image format, embedding malware inside a PostScript file could enable that to be run without user interaction, as with other graphics formats.

There are three major PostScript interpreters in common use:

- *Adobe's Distiller engine, built into its Acrobat products,*
- *Apple's PSNormalizer engine, built into macOS,*
- *Artifex's open source engine, built into Ghostscript, and widely used in Linux and other platforms.*

Research into those engines has so far been relatively limited, but has revealed some serious vulnerabilities. Most recently, Kai Lu and Brett Stone-Gross of Zscaler's ThreatLabz published an account of three vulnerabilities they found in Distiller, and one in PSNormalizer, in 2022. Those were fixed by Adobe and Apple last year, in the latter case in macOS Monterey 12.5 released on 20 July 2022, and its equivalent security updates for Big Sur and Catalina. From the dates, I suspect that the removal of support for inline viewing of PostScript files in WebKit

in Monterey 12.3 may also have been part of Apple's mitigations.

Apple was most probably prompted into conducting a security audit of PSNormalizer as a result of the vulnerability reported, and would have been faced with the choice of re-engineering it or removing it from macOS completely. Unlike the PDF engine in Quartz, PSNormalizer is now little used, and has no significant role in macOS. The first step was to make it inaccessible from the GUI by disabling that feature in Ventura's Preview, then following that in Sonoma by removing PSNormalizer altogether, so removing its command tool pstopdf and Core Graphics' CGPSConverter.

This leaves those still wishing to convert PostScript files with a choice between Adobe's Distiller in its paid-for Acrobat products, and Artifex's Ghostscript, which has had its own share of vulnerabilities. There is also a third option, of running a late version of macOS Monterey in a lightweight VM and continuing to use PSNormalizer through Preview there. For most, that will be the cheapest and simplest option.

John Warnock, co-founder of Adobe and driving force behind PostScript, died on 19 August this year. His page description language had brought success to the Mac for almost 39 years, but at least PDF lives on.

For the sake of completeness I'll note that this author added a (ahem) "Postscript" to his posting, writing:

I'm now starting to see warnings from third-party app developers that, in Sonoma, their apps will be unable to open or import EPS files as they can no longer convert them using Quartz in macOS 14. Although the impact on most of us should be very small or negligible, if you do still use EPS or PostScript at any scale, you will need to prepare a solution for continuing to do so after upgrading to Sonoma. I wish you success.

https://eclecticlight.co/2023/09/25/postscripts-sudden-death-in-sonoma/

To Apple, I say "bravo." It is always difficult to kill off features that have any audience. And Apple takes some heat whenever they decide to break the status quo in the interest of a better future. Postscript is a big, old, and very dangerous interpreter. There's no doubt that some people will complain. That's inevitable. But the fact that something that's inherently dangerous **could** be removed with relatively little repercussion suggests that Apple once again made the right call.

The NSA hacked Huawei? … (Yeah, in 2009)

Last Tuesday China's Ministry of State Security published an extremely rare (as in the first time ever) official statement on its WeChat account. It formally accused the US National Security Agency of hacking and maintaining access to servers at Huawei's headquarters since 2009. What's interesting is not that this is news, it's that it's not news. So I'm mentioning this because this reflects a change in stance for China and this might just be the beginning. The question is, the beginning of what? Both the New York Times and Der Spiegel originally reported this back in 2014 based upon documents from the Snowden leaks which disclosed "SHOTGIANT", an NSA operation to compromise Huawei's network for the purpose of finding links between Huawei and the Chinese PLA, to learn Huawei's internal corporate structure, and identify ways to exploit Huawei equipment—which, at the time, was being widely adopted by both US allies and

adversaries. This is the first time the Chinese government has ever publicly confirmed the NSA's Huawei hack. And they posted it on their WeChat channel.

The Chinese Ministry of State Security statement doesn't go into any technical details about the actual hacking. It just recycles information published by the New York Times and Der Spiegel and the Snowden leaks. It does spend a lot of time accusing the US of using (and I'm quoting) *"the despicable tactics of the 'Matrix' to maintain a 'cyber hegemony'."* To that end, Chinese officials claim that the **U.S.** is doing all of the intellectual property stealing and then using its allies and PR machine to hype, exaggerate, and smear China on the *"Chinese cyber secret stealing issue."*

A cyber threat analyst at the Taiwan-based cybersecurity firm TeamT5 was quoted saying:

> *"Considering the close relationship between China's cybersecurity firms and the Chinese government, our team surmises that these reports could be a part of China's strategic distraction when they are accused of massive surveillance systems and espionage operations."*

There have been several other recent reports of NSA penetration into China's space. I suppose none of us assumed that the cyber intrusions were all one-way. We have an NSA and all of those people dressed in 'cammo' must be up to something. It seems as though the Chinese government may have changed policy. Rather than pretending to be invulnerable, they've decided that the better strategy is to acknowledge that the U.S. is also intruding into Chinese affairs. This might also be a reaction to China's very high profile and heavily publicized intrusion into Microsoft and their exploitation of their access to enterprise eMail. There does appear to be a difference, though. The evidence we have suggests that when we get into their networks we just snoop around to gather intelligence. When they get into our networks proactive damage results.

As we recently saw, that Chinese attack on Microsoft took a great deal of effort. That demonstrated some serious cyber skills. But sometimes you just trip over a pot of gold. That was the case when a misconfigured Azure Shared Access Signature (SAS) token resulted in 38 terabytes of hyper-sensitive Microsoft data being exposed for the taking – and not only for the taking...

Microsoft's big AI data blunder

A cloud security focused group, known as Wiz Research (that's not "Whiz" – there's no 'h' – just Wiz, as in Wizard) they stumbled over a trove — and I mean "trove" as in "we're going to need a bigger drive over here" — trove of Microsoft data, all exposed and sitting there out on the Internet. The Wiz wizards explained:

> *As part of the Wiz Research Team's ongoing work on accidental exposure of cloud-hosted data, the team scanned the internet for misconfigured storage containers. In this process, we found a GitHub repository under the Microsoft organization named* ***robust-models-transfer****.*

(It sounds like maybe the models were robust but not the security.)

The repository belongs to Microsoft's AI research division, and its purpose is to provide open-source code and AI models for image recognition. Readers of the repository were instructed to download the models from an Azure Storage URL.

Oh, so it was deliberate? That doesn't sound bad. But no... were getting to the bad part:

This URL allowed access to more than just open-source models. It was configured to grant permissions on the entire storage account, exposing additional private data by mistake.

Our scan shows that this account contained 38TB of additional data — including Microsoft employees' personal computer backups. The backups contained sensitive personal data, including passwords to Microsoft services, secret keys, and over 30,000 internal Microsoft Teams messages from 359 Microsoft employees.

... And that's where you say *"Whoopsie!"* It also occurs to me from this report that you don't know that there are 30,000 internal Microsoft Teams messages from exactly 359 different Microsoft employees without doing a great deal of data analysis and counting things. So, yeah, they probably did need a bigger drive. The report continues to explain…

In addition to the overly permissive access scope, the token was also misconfigured to allow "full control" permissions instead of read-only. Meaning, not only could an attacker view all the files in the storage account, but they could delete and overwrite existing files as well.

Okay. Now wait a minute. This really does cause us to question the use of the term "attacker" here. If you pick up a lost USB thumb drive in a parking lot, have you attacked anyone? It seems to me you're just … observant. So if Microsoft is waving their arms around and saying: *"Hey, come over here and download a bunch of stuff using this URL! ... oh, and while you're here, how would you like to read 30,000 pieces of private internal corporate communications from 359 of our employees?"* Does it qualify as an attack if you say "Well thank you, that does sound interesting!"

This is particularly interesting considering the repository's original purpose: providing AI models for use in training code. The repository instructs users to download a model data file from the SAS link and feed it into a script. The file's format is ckpt, a format produced by the TensorFlow library. It's formatted using Python's pickle formatter, which is prone to arbitrary code execution by design. Meaning, an attacker could have injected malicious code into all the AI models in this storage account, and every user who trusts Microsoft's GitHub repository would've been infected by it.

However, it's important to note this storage account wasn't directly exposed to the public; in fact, it was a private storage account. The Microsoft developers used an Azure mechanism called "SAS tokens", which allows the the creation of a shareable link granting access to an Azure Storage account's data. This means that upon inspection, the storage account would still seem completely private. But as we now know, it was anything but.

In Azure, a Shared Access Signature (SAS) token is a signed URL that grants access to Azure Storage data. The access level can be customized by the user with permissions ranging from read-only to full control, while the scope can be either a single file, a container, or an entire storage account. The expiration time is also completely customizable, allowing the user to create never-expiring access tokens. This granularity provides great agility for users, but it also creates the risk of granting too much access; in the most permissive case — as was the case with Microsoft's token above — the token can allow full control permissions, on the entire account, forever – essentially providing the same access level as the account key itself.

So, we can be glad, or at least hope, that adversarial cyberattackers didn't stumble upon this. We can be glad that the Wiz Research guys did and that they promptly (after doing a bunch of counting to see how much of what was there) gave Microsoft a heads-up about this little misconfiguration mistake.

Microsoft's own MSRC blog posting for this incident was titled: *"Microsoft mitigated exposure of internal information in a storage account due to overly-permissive SAS token."* Right. I won't spend any more time on this other than to note that in Microsoft's sharing of their "learnings" (yes, they did call it that) they never got around to mentioning just how much of their highly sensitive data had been found flapping in the breeze.

Signal's PQXDH Quantum-Resistant Encryption

The Signal encrypted messaging platform was originally named "Axolotl" after the newt which has self-healing powers. In some ways Signal, which uses a resynchronizing cryptographic ratchet system, is a similarly self-healing protocol. It was originally designed by Moxie Marlinspike of Whisper System, and then wisely renamed from "Axolotl" to "Signal." And it hit the big time when this protocol was adopted by Meta as the secure messaging protocol for their WhatsApp messenger. If the phrase "resynchronizing cryptographic ratchet system" doesn't ring and bells, if you're a listener who joined us after April 12th of 2016, or you're a longtime listener who would be interested in a refresher. I did one of our famous deep dives into this truly lovely messaging protocol which, thanks to Meta's adoption, is now in use by more than one billion people worldwide. Look for Security Now! episode #555 which we titled "WhatsApp."

The reason we're talking about Signal today is that it Signal, the company, made headlines last week in the tech community with their announcement that their already extremely clever and well-designed Signal protocol was being upgraded with something they called **PQXDH** quantum resistant encryption. If I didn't already have a topic for today's podcast, this would have been it. But we have room for both.

Signal's move might at first appear premature, since the threat posed by quantum computers to the public key crypto we rely upon today, remains purely theoretical and may well remain so for the foreseeable future. We've had some fun at quantum computing's expense, noting that breaking RSA-style crypto, which would require determining the prime number factors of a massive number represented by more than binary 4000 bits, appears to be safe for now since it was considered to be a breakthrough and huge accomplishment when today's most advanced quantum computer successfully factored the number 35.

Given that, it might appear that Signal's move to an overtly quantum resistant protocol is premature. If nothing else, it's at least brilliant marketing. But there's more to it than that. The security world has coined another abbreviation which is pronounced "Handle" because the abbreviation is H.N.D.L. HNDL stands for "Harvest Now, Decrypt Later." And we know that our dearly beloved NSA has built a truly massive 1 to 1.5 million square foot data center, of some sort, out in the boonies of Utah. So there's more than a passing chance that the "Harvesting Now" first half of the Harvest Now, Decrypt Later strategy is already well underway.

In other words, if you want your secrets to remain secret past the foreseeable future, it's never too soon to begin encrypting under post-quantum crypto technology. That's what makes Signal's announcement last week significant. So what exactly has Signal done?

Signal's current shared secret key agreement protocol is known as X3DH. The "DH" is short for Diffie-Hellman which is a well-established key agreement system. We've discussed key agreement protocols in general, and Diffie-Hellman in particular, many times in the past. And I know it pretty well since it plays a large part in SQRL's more tricky security features. Briefly, a key agreement protocol allows two ends of an insecure and public connection to exchange some information in plain sight while each obtaining a shared secret. It seems counter-intuitive that their conversation could be completely known, while they each arrive at the same secret that only they know – but it works. The "X3" refers to the "X25519" elliptic curve which is the flavor that Signal's Diffie-Hellman key agreement has traditionally used. And what's really cool is that they're still going to in the future.

What they decided to do is to **add** another key agreement protocol – this one believed to be quantum-safe – **to** their existing system in such a way that BOTH of the protocols, the old and reliable elliptic curve Diffie-Hellman, and the "believed to be safe today and tomorrow" new flangled quantum-safe system, both both need to be simultaneously broken and cracked in order for an attacker to obtain the shared secret that's used to encrypt Signal's communications.

Signal selected one of the NIST-contest finalists that's believed to be the best, known as CRYSTALS-Kyber. We've talked about it previously. They chose it because it's built upon a solid foundation. But they also wisely decided that since it's still new and unproven, they would not depend upon it solely. So Signal's original X3DH has been renamed PQXDH – PQ for post-quantum and XDH for its elliptic curve Diffie-Hellman which survives.

And get this: This new PQXDH is already present and supported in the latest versions of Signal's client applications and it's already in use protecting any conversations initiated after both sides of the chat are using the latest Signal software which supports both the original and the updated PQXDH. Then, in the future, after sufficient time has passed for everyone using Signal to have updated, they plan to disable the use of the old non-quantum-enhanced X3DH for all new conversations and to then require PQXDH for all new chats.

It's clear that we're moving into a post quantum world. Now that Signal has led the pack by introducing quantum-safe messaging, the rest of the pack, who may have been caught by surprise, will have no choice but to figure out how to do the same, or be quickly left behind.

Closing the Loop

Dustin Smith / @GIDustin

Our listener, Dustin Smith, put me onto a blog posting which was originally written three and a half years ago on March 18th of 2020 by a guy name Drew DeVault. It's an on-point rant about what modern web browsers have become and I think everyone here will find it interesting. Here's what Drew wrote:

Since the first browser war between Netscape and Internet Explorer, web browsers have been using features as their primary means of competing with each other. This strategy of unlimited scope and perpetual feature creep is reckless, and has been allowed to go on for far too long.

I used wget to download all 1,217 of the W3C specifications which have been published at the time of writing (not counting WebGL, which is maintained by Khronos). Web browsers need to implement a substantial subset of this specification to provide a modern web experience. I ran a word count on all of these specifications. How complex would you guess the web is?

The total word count of the W3C specification catalog is 114 million words at the time of writing. If you added the combined word counts of the C11, C++17, UEFI, USB 3.2, and POSIX specifications, all 8,754 published RFCs, and the combined word counts of everything on Wikipedia's list of longest novels, you would be 12 million words short of the W3C specifications.

I conclude that it is impossible to build a new web browser. The complexity of the web is obscene. The creation of a new web browser would be comparable in effort to the Apollo program or the Manhattan project.

It is impossible to:

- *Implement the web correctly*
- *Implement the web securely*
- *Implement the web at all*

Starting work on a bespoke browser engine with the intention of competing with Google or Mozilla is a fool's errand. The last serious attempt to make a new browser, Servo, has become one part incubator for Firefox refactoring, one part playground for bored Mozilla engineers to mess with technology no one wants, and zero parts viable modern web browser. But WebVR is cool, right? Right?

The consequences of this are obvious. Browsers are the most expensive piece of software a typical consumer computer runs. They're infamous for using all of your RAM, pinning CPU and I/O, draining your battery, etc. Combined, web browsers are responsible for more than 8,000 CVEs alone.

And then Drew switches from establishing some facts into making a very interesting observation. He writes:

Because of the monopoly created by the insurmountable task of building a competitive alternative, browsers have also been free to stop being the "user agent" and start being the agents of their creators instead. Firefox is filling up with ads, tracking, and mandatory plugins.

Chrome is used as a means for Google to efficiently track your eyeballs, and muscle their anti-technologies like DRM and AMP into the ecosystem. The browser duopoly is only growing stronger, too, as Microsoft drops Edge and WebKit falls well behind its competition.

The major projects are open source, and usually when an open-source project misbehaves, we're able to fork it to offer an alternative. But even this is an impossible task where web browsers are concerned. The number of W3C specifications grows at an average rate of 200 new specs per year, or about 4 million words, or about one POSIX every 4 to 6 months. How can a new team [of any forked browser project] possibly keep up with this on top of implementing [and maintaining] the outrageous scope web browsers already have now?

The browser wars have been allowed to continue for far too long. They should have long ago focused on competing in terms of performance and stability, not in adding new web "features". This is absolutely ridiculous, and it has to stop.

https://drewdevault.com/2020/03/18/Reckless-limitless-scope.html

I think the only thing I would change about what Drew wrote is his last line, which I would change to read: *"This is unfortunate in the extreme but there doesn't appear to be any way for the industry to change that course."* And I would further add that, to their credit, Microsoft perceived very early on just how important the web browser would be to the future. So they attempted to build it into Windows and called it "Internet Explorer." As we know, what the web has become even outgrew **their** ability to browse it. Once upon a time, IE was the most widely used browser, attaining a peak of 95% share 20 years ago in 2003. This podcast began two years later and we all witnessed IE's decline and its inevitable death. After making a massive investment in a brand new post-IE engine Microsoft, to their credit, recognized that a modern web browser was too big for even them and that they'd be better off just hanging bells and whistles off a browser that someone else maintained and evolved.

Are there any takeaways for us here? Nope. I can't think of any. Drew's observations, which I think are valuable, pair beautifully with what I've noted to be true of operating systems. Just as no one can really create a competitive and useful truly new operating system from scratch any longer, any operating system that a user is going to sit in front of will need to host a web browser that is all by itself also too massive and complex to create from scratch.

I think all of this was absolutely inevitable. And that further suggests it wasn't a mistake, it's not a consequence of inertia (which we often observe elsewhere) and we're almost certainly going to continue moving down this path we're on.

Anonymous / @Anon_Sickness

Hello Mr. Gibson, I hope I can ask you a question, over the last podcast from Security now. Around 30 min, you say that you encrypt with the private key and decrypt with the public key. Is this right? I think it's reverse. What I learned is that you encrypt with the public key and decrypt with the private key. Is this right or I have wrong information? Thank you for response.

One of the powerful beauties of public key cipher technology is that the process works in either direction. It's obviously very useful to encrypt something using someone's public key that you know only they will be able to decrypt with their secret private key. PGP is a perfect example of this. And digital signatures of all kinds is the best example of this being done in reverse. In fact, PGP does this, too. To provide authentication of the sender of a message, the sender uses their private key to sign the file which the recipient is able to verify using the sender's public key. When the recipient is verifying the signature they're decrypting a hash of the document that the sender encrypted with their private key.

The last time we talked about public key technology I was pleased with the way I conceptualized the public and private key distinction used by pre-elliptic curve RSA crypto. You start by obtaining a private key which is just a really large prime number. And remember that counter-intuitive though it is, prime numbers do not become more scarce as we go farther and farther out. There's always plenty of them. Okay... so we have a private key.

Now, we want to arrange to hide that private key inside the public key. We do this by taking another very large prime number and multiplying that new prime number with the original private key which was also a prime number. Though I've simplified things, that multiplied key is the public key. The public key contains the private key but there's no way to know what it is after they've been multiplied together because to this day, despite all of the best brains in math – and I mean this has received a lot of attention through the years – no one has ever come up with a practical way to "unmultiply" those two primes through the process of prime factorization. And, by the way, **that** is the danger posed by quantum computing. There are reasons to believe that some future quantum computer might be able to finally factor a very large public key back into its original two prime numbers, one of which would be that public key's matching private key. Thus the private key would be revealed and could be used.

The true magic of public key crypto is that the private key, which has been hidden inside the public key, is still able to perform encryption and decryption despite being completely hidden by its multiplication by another large prime number. It's all so cool.

Jerry *Here's a particularly interesting one...*

> *Your assertion that "no deterministic mathematical algorithm can generate a random result." is incorrect. Non-deterministic mathematical algorithms (processes) can generate true random results, not "pseudo-random" numbers. I developed such an algorithm.*

Okay, so first of all, I said "no deterministic mathematical algorithm" not "non-deterministic mathematical algorithm". The problem is that the phrase "non-deterministic mathematical algorithm" is, as far as I know, an oxymoron. All mathematical algorithms are inherently deterministic. If I made a mistake on my math homework in elementary school, explaining to my teacher that this was my implementation of a non-deterministic mathematical algorithm, I'm pretty certain that would not have gone over very well, nor would it have changed my grade.

So, Jerry, if you have indeed developed a non-deterministic mathematical algorithm which is capable of generating true random results, there's probably a Nobel Prize waiting for you in Stockholm. And more importantly, what you may have actually done is uncovered a provable

flaw in the simulation we're all living within. But seriously, you've got my attention, you have access to an audience... are you going to just leave us all hanging like that?

Now, in a bizarre coincidence, just as I was getting ready to cast these show notes from Google Docs where I write them every week into their PDF form, I happened to glance down at my phone and see a reply Tweet from this person. Yesterday when I encountered his assertion, not knowing how to respond to something which clearly seemed impossible, I simply replied *"I hope you've submitted this for a Nobel Prize!"* He didn't take that very well. Certainly not in the somewhat carefree spirit it was meant. So the first thing I did was to anonymize his identity in the show notes since I certainly don't want to embarrass him from this dialog. What he Tweeted in reply to my short comment about the Nobel Prize (and actually I'm kind of serious about that) was the following. He said:

> *I'm very surprised by your reply. I've been a customer and promoter of your products and services for 30+ years. I have read numerous articles of yours. I never expected sarcasm from you. My bad. I developed these, and similar, algorithms more than 45 years ago. I have always expected others to develop like algorithms, but to date this has not occurred.*

Now, again, I'm having a little bit of trouble taking this seriously or at least as seriously as he is. When he says: *"I have always expected others to develop like algorithms, but to date this has not occurred."* My first impulse is to say "Gee... I wonder why that is?" Instead, this what I replied in turn:

> *Hi Jerry.*
> *Okay. So I suppose my reply was somewhat sarcastic. And I certainly would love to be proven wrong in my assertion that (to quote you quoting me) "no deterministic mathematical algorithm can generate a random result." The trouble is, I still believe that to be true. And not just a little bit true, but* ***very*** *true. If you're a mathematician then you understand that the way to proceed from here would be to offer up some proof of your assertion, which appears to defy the laws and principles of mathematics and reality. I'd be more than happy to eat my sarcasm if you can demonstrate that I'm wrong! :)* [exclamation point, smiley face]

If I do hear anything more from Jerry I'll let everyone know.

Todd / @austincableguy

> *Hey Steve, I listened to Episode 940 where you explain securely wiping drives by writing random bits to all sectors of spinning drives. Lately, I have been using Veracrypt to encrypt old drives before I get rid of them (using GRC's password page to generate a secure PW). Is this as secure as your upcoming software that writes random noise?*

That's pretty clever. If you did it twice under different keys then it would be the equivalent of writing random noise as GRC's solution will. And I wouldn't argue that that second pass is absolutely necessary. I might offer a single-pass option for those in a hurry. VeraCrypt, like its ancestor TrueCrypt, uses XTS cipher mode encryption, which is the industry standard and NIST-approved way of encrypting block-addressed mass storage using what's known as a

tweakable cipher. And we also know that any good modern cipher, like AES, generates pseudo-random noise that's indistinguishable from true random noise. It'll be slower than GRC's product, since it's reading and writing. And I don't know how large the blocks are that it's using. It won't be able to access and wipe the latent data that might exist in grown-defect spared out sectors. But it's a nifty solution until GRC's product is ready. It's here today and it's free. So thanks for sharing the idea. It's pretty clever and a great application. I can't think of any downside.

Bill Melvin / @billmelvin Bill had a slightly different take on random noise wiping

Hi Steve. I have a thought regarding the discussion on wiping hard drives on show 940. I assume TrueCrypt/VeraCrypt hidden volumes look like random noise. It seems to me that doing an extra pass writing zeros to a drive takes you from "plausible deniability" to absolute impossibility. In an abundance of paranoia, why take the risk of being badgered over something that is not there. ;-) Thank you for Security Now, which I have enjoyed for years. I have my SpinRite license and am looking forward to the new version.

So Bill's point is that leaving a drive **looking** like it contains encrypted data might be more troublesome than leaving it looking empty. I think that's a good point. I mentioned that briefly last week as an option though not for the reason Bill mentions, which is a good one.

mark kozel / @MarkKozel

I found your description of SSD wear-leveling fascinating. It is amazing what engineers have come up with to handle issues like this. It got me wondering... I have a portable SSD with a single multi-gigabyte VeraCrypt blob on it. I open the blob a few times a week and make changes to files inside. After closing the blob everything appears to be unaltered, as far as Windows knows. I set VeraCrypt to change the file mod date on the blob when I realized syncthing needed that to detect a change and sync the blob to my NAS. Does the wear-leveling capability work on the blob? Is it active inside the blob when I have it open in VeraCrypt? Love the show (and continuing education credits I get towards my Security+ certification by listening)

So the answer is: Yes. The wear-leveling is occurring at the lowest possible level in the chain of data flow. It is not visible to anyone reading or writing to and from the drive, so it functions whether the data being read or written is encrypted or not. And one slick benefit of using VeraCrypt to encrypt the blob on the SSD is that since VeraCrypt always leaves the data on the SSD encrypted – only decrypting it on the fly – as wear leveling causes blob updates to be written to dispersed regions of the drive, thus leaving obsoleted bits of the blob behind, those obsoleted and isolated blob bits are also encrypted. So **that** drive will never need to be wiped, at least not to protect anything that was ever stored in the blob. Once VeraCrypt's volume header has been overwritten multiple times, there is no way to ever recover any of that blob's data. Period.

Wayne Patton / @GeekoHog

How difficult would be to change the signed integer to an unsigned something else and recompile and fix Unix time.

I should have – but failed – to mention that this UNIX time problem was fixed in the Linux Kernel back in June of 2018 after two and a half years of research into the right way to do it without breaking anything else. In the process, more than 80 source files hade to be changed. So this Y2038 problem is no longer any concern for Linux kernels since then. The worry is that before this, all embedded Linux kernels of the sort used for industrial automation, satellites, elevators, and who- knows-what else, will have this trouble and in those markets there's much more of a "if it's not broken don't mess with it" mentality. So the concern is not that mainstream Linux or any of the various major UNIXes will break. They've all been fixed. It's that some satellite that's been in orbit since before this was fixed may become confused and do who knows what? Or maybe a long forgotten pump on an oil rig. Anyway, it's going to be interesting to see whether it's another non-event like Y2K, or whether something interesting occurs.

B0wter / @b0wter

Hello Steve, Thank you for the many great episodes of Security Now. I've just listened to episode #939 and have a thought about what you said regarding the LastPass hack. You mentioned that hackers will go after those accounts where they suspect a big payout. I think you're right about that, but I also think we must not let our guard down. All accounts will eventually be decrypted, and all accounts will be available for purchase on the dark web. This will be a problem for all those (including me) who used LastPass to store information like social security numbers and other personal IDs. I think it's important to take precautions while we still can. Kind regards, Johannes.

I wanted to share Johannes' note because I understand his concern and I imagine it's shared. But I also doubt that the vast majority of the more than 25 million users whose encrypted data was stolen, are in any real danger. Even the advent of quantum computing won't make any difference since we're talking about brute-forcing symmetric crypto which quantum computing does not threaten. The cost of performing 100,100 rounds against a hopefully high-entropy password is truly prohibitive. These crooks appear to be spending significant money on cracking gear or cloud compute cycles, and they appear to be cracking only a couple of vaults per week. That's perhaps a hundred or so per year against a population of more than 25 million.
It's true that someday they might eventually get around to opportunistic attacks against non-high profile targets whose iteration counts are low. But even then, if the passphrase was good, and there's no clear reason to believe that any actual cash lies on the other end of the crack, it still takes a significant investment of time and money to turn someone's vault back into plaintext. It really isn't ever going to be easy.

I can't tell anyone that there's absolutely nothing to worry about. So my intention is only to present some probably-accurate perspective. Cracking these vaults will never be free and these people are probably not ones to throw away good money.

Kevin van Haaren / @kvanh

Listening to the LastMess episode I realized I had some info in my LastPass vault in addition to passwords, things like ssh private keys, recovery keys for all Apple devices and various cloud services. I didn't even think of rotating all those when I moved off LastPass. I'm also thinking some of that shouldn't be in my LastPass replacement. ssh keys I'll probably keep on an

encrypted disk image on my computer instead. Not sure on recovery keys, i may adopt your 2FA authenticator app and just print out recovery keys and keep them some place safe.

I wanted to share this because I think Kevin's observation that just because today's password managers offer synchronized encrypted vaults, doesn't mean that they are the best place to store unrelated non-web secrets. Password managers need our usernames, passwords and credit cards; and perhaps some additional automatic form-fill information for convenience. But things like SSH keys, server certificates, and other unrelated non-web passwords don't all need to be stored in a single place. Yes, it's convenient. And that convenience can be seductive. But when you combine the wisdom about not keeping all of our eggs in one basket and the old "fool me once..." with LastPass being that once... and when we consider that today's computing environment does not have any more dangerously exposed – and complex software, as we saw above – than our web browsers, it would be difficult to make a case for storing more in our browsers than is needed for web-facing activity.

I think a good cross-platform encrypted archiving program is probably a good solution. It can contain all kinds of stuff, it squeezes it down, and then any file synchronization tool, any of the cloud drives or 3rd-party sync systems like SyncThing, can keep the copies synchronized. I'll be interested to hear if any of our listeners have a better idea.

Don K / @DonK358

@SGgrc Thank you Steve. You are the only reason I still use this Twitter or 'X' account, so I can now look forward to deleting it as soon as your email solution arrives.

…And…

Mark Newton / @Lmarkn

I am one of the few people who may hold out a little while and hope Elon turns this thing around. The boy does have a track record of success. Dropping the name Twitter was almost as dumb as buying the thing. I will give him another year. It will either be turning around or Twitter/X will be a thing of the past.

A few people took exception to what they called my "anti-Elon anti-Twitter rant" last week. Others confirmed what I suspected, which was that the only reason they still had Twitter was to easily receive my weekly Security Now! tweets. So allow me to clarify just a bit: I'm currently paid up for a year, since I can't survive Twitter without Tweetdeck. And I'm fine with that. You didn't see me budge one inch when everything that has happened so far, happened. I'm also in no big hurry to leave. My sole concern was that a switch to a 100% paid model – which Leo thinks may never happen, and I understand that, too – would be a profound change. I'm already paid up, so it makes no difference to me. But if there are people, and I've heard from some, who are only on Twitter to obtain my weekly Tweets, then I disliked the idea of **any** of our listeners feeling that they needed to go "paid" just to continue to receive that feed. I wanted those people to know that there would be an alternative means for obtaining that information.

GRC also has no means at the moment for reaching any community outside of Twitter. So as I've

mentioned before, I have planned to remedy that once SpinRite is released. Since I'll have the ability to mail to lists, it makes sense to have a list for the Security Now! Podcast.

I'll be happy to keep tweeting the weekly show note links, the picture of the week and a brief synopsis of the show's topics. But it no longer seem right for Twitter to be the only way for our listeners, of which there are many, to receive that information.

Miscellany

Konrad Zydroń / @KonradZydron

> *About that HashCheck SHA-256 fork you mentioned in SN940, it's here since 2014:* https://github.com/gurnec/HashCheck
> *Another fork with additional BLAKE3 algorithm and faster implementation of SHA-256, SHA-512 and SHA3*: https://github.com/idrassi/HashCheck

The second fork that Konrad mentions is the one we want since the developer who forked it said that SHA-256, -512 and SHA3 received tremendous speed improvements courtesy of OpenSSL. The links are in the show notes and I created a GRC shortcut: HashCheck.
So: https://grc.sc/hashcheck which will take you to the Github release downloads page.

When Hashes Collide

And, finally, it's safe to say that last week's podcast topic "When Hashes Collide" made one library information scientist very happy. Ray wrote:

> *Hi Steve! I was absolutely floored while listening to the show on Friday morning to hear that my question was turned into a truly fascinating, and **VERY** useful deep-dive!*
>
> *Our patron population is a little bit bigger than 5,000 individuals—it's actually closer to 400,000. I did say "a large public library system in Ohio" and Ohio is known to be the "Land of Libraries"—at least in certain circles! But the topic you presented is still of course very much applicable.*
>
> *The idea of deliberately using the Birthday Paradox and further obscuring personally identifiable information (PII) by manipulating the number of bytes produced by the cryptographic hash is an absolutely brilliant way to add a meaningful layer of protection to the pseudonymized data! Controlling the probability of collisions within the population of patrons is something I didn't even realize I was looking for to give a small degree of uncertainty to the pseudonyms! Once again, an absolutely fantastic explanation of such a complex topic! Bravo!*

It's a bit embarrassing to read that, but thank you, Ray. What most pleases me is that Ray "got it" completely and has already turned the concept into working code. He calls it the "Stochastic Pseudonymizer" and for anyone who's curious I have the link to his Github project in the show notes.
https://gist.github.com/rayvoelker/598172d06bcc94c51dfa8064bbb0cb75#file-stochasticpseudonymizer-ipynb

We told you so!

Last Thursday, Apple quickly patched three iOS vulnerabilities. Taken together they created a 0-day chain that enabled a potent attack that The Citizen Lab in Canada and Google's TAG team (their Threat Analysis Group) had jointly reverse-engineered from the forensic evidence residue left behind by a successful targeted attack aimed at an Egyptian citizen who had made himself a political target by announcing his intention to run in a forthcoming election against Egypt's current head of government.

I named this episode **"We told you so!"** in acknowledgement of our many listeners who took exception to my assertion that not all websites in the world needed HTTPS. What Citizen Lab and the TAG team discovered was that it was when this political target was led to a plain HTTP, non-HTTPS website, that his network traffic was intercepted on-the-fly by a man-in-the-middle middlebox. That device returned an HTTP 307 Temporary Redirect which caused his iPhone to be redirected to a spoofed website where the malicious content could be injected into his phone via this chain of three carefully designed exploits. So, indeed, no argument... **you told me so**.

Let's take a closer look at some of the interesting details here...

The target of the attack was Ahmed Eltantawy, a former Egyptian Member of Parliament. Earlier this year in March he announced his intention to run in the upcoming Egyptian presidential election, stating that he planned to offer a "democratic" alternative to the current president. Following this announcement he, his family members, and supporters have been subjected to harassment including reported arrests of 12 family members.

Egypt's current president Abdel Fattah el-Sisi has been in power since 2014, when he led the military overthrow of President Mohammed Morsi. Sisi has been widely described as an autocrat. Human rights groups, including Amnesty International and Human Rights Watch, have documented widespread human rights abuses under el-Sisi's regime, including repression against civil society groups, activists, and political opposition. So, Egypt's current president doesn't appear to be a nice guy who plays fair or would be in favor of free and fair elections.

Ahmed became suspicious about the safety of his phone and reached out to Citizen Lab, who then performed a forensic analysis of his device. Their analysis revealed numerous attempts to target Ahmed with **Cytrox's Predator spyware**. Citizen Lab has previously documented Cytrox Predator infections targeting the devices of two exiled Egyptians, an exiled politician and the host of a popular news program (who chose to remain anonymous).

Citizen Lab brought Googe's TAG team into the project to perform some of the heavy reverse engineering forensics which the TAG teams excels at. This allowed them to obtain the complete iOS exploit chain that had been targeted at Ahmed. They initiated a responsible disclosure process with Apple, which assigned three CVEs to vulnerabilities associated with the chain:

CVE-2023-41991 (Security): A malicious app may be able to bypass signature validation.
CVE-2023-41992 (Kernel): A local attacker may be able to elevate their privileges.
CVE-2023-41993 (WebKit): Processing web content may lead to arbitrary code execution.

The zero-day chain was hosted on sec-flare[.]com, and the exploit also contacted verifyurl[.]me. After fingerprinting those two websites they scanned the Internet and identified a large number of IPs that matched those sites' fingerprints. In other words, there were other domains that were pointing to the same malware hosting sites and they considered all of those IPs (and the domain names returned in TLS certificates when they matched the fingerprints) to be directly linked to Cytrox's Predator spyware.

They also observed that some of the domains they had identified had names suggestive of tailoring towards specific countries or regions of including the Arabian Gulf, Southeast Asia, Angola, the Democratic Republic of the Congo, Egypt, Greece, Indonesia, Kazakhstan, Madagascar, Mongolia, the UAE, and Sudan.

By examining the final stage of the iOS exploit chain, which was an iOS payload, they were able to attribute, with very high confidence, that this was Cytrox's Predator spyware which they had obtained another sample of back in 2021. The two binaries shared a key similarity which could only be accounted for through a common heritage.

Okay. So what about this on-the-fly interception?

This month of September and last month, when the target, Ahmed, visited certain websites without HTTPS from his iPhone, using his Vodafone Egypt mobile data connection, he was silently redirected to a website at the domain "c.betly[.]me" via a on-the-fly network injection. And that domain was among those that matched Citizen Lab's fingerprint for Cytrox's Predator spyware.

The injection was triggered by a string match on the website specified in the HTTP Host header, as well as the value of the User-Agent header. I'll note that while the target domain can also be seen at the start of a TLS handshake thanks to SNI (Server Name Identification), subsequent TLS encryption blinds anyone eavesdropping from seeing additional connection details such as the User-Agent header. Since HTTP leaves everything in the clear the attackers can use these additional signals to reduce mis-targeting since they would not want non-targeted web traffic to be redirected.

So, as I said before, when the injection attack is triggered, the intercepting middlebox immediately returns a 307 Temporary Redirect, which will be invisible to the user, and it also blocks the legitimate reply which would have been returned from the authentic server.

The body of the page returned from the malicious redirected website included two iFrames. One which they described as containing apparently benign bait content – it was a link to an APK file not containing any spyware – but the second was an invisible iFrame containing a Predator infection link hosted on sec-flare[.]com.

Next, they wanted to understand where in the network the injection was occurring. So they performed a bunch of tricky IP packet logic to localize the malicious middlebox. They needed to get far more sneaky than just using traceroute-style TTL (time to live) manipulation because they discovered that another benign traffic management middlebox was in the path and located close to Vodafone subscribers. But they were eventually able to determine that the malicious

middlebox was on the interconnect between Telecom Egypt and Vodafone Egypt... which is exactly where you'd expect it to be if you wanted to be able to reliably intercept and infect targeted smartphone users. They were also able to identify the specific hardware by probing it remotely and comparing its responses to other known appliances. It's a commercial off-the- shelf (COTS) device made by Sandvine with offices in San Jose, CA, Sweden, Ontario Canada and India. It's one of their PacketLogic devices: https://www.sandvine.com/products/packetlogic

Sandvine's data sheet boasts features like:

- POLICY-BASED TRAFFIC MANAGEMENT CAPABILITIES that include asymmetric traffic control, traffic shaping and filtering, traffic flow classification and prioritization, traffic monitoring, **and packet rewrite.**

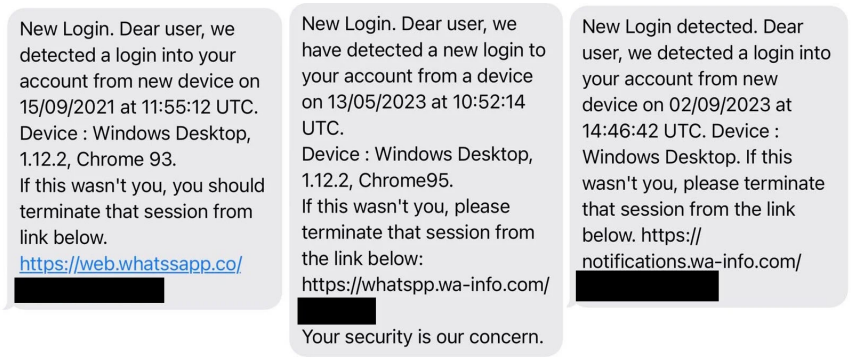
- CONTENT INTELLIGENCE enables network operators, cloud providers, and high-speed enterprises to combine the policy management capabilities of PacketLogic with industry-leading URL content categorization functionality.

- HTTP HEADER ENRICHMENT leveraging PacketLogic's stateful awareness and subscriber awareness.

- ADVANCED TRAFFIC STEERING that combines PacketLogic Real-Time Enforcer capabilities with Application Delivery Networking functionality to provide a single Application Delivery Controller solution with unmatched performance and scalability, enabling service chaining with subscriber, service plan, charging, and Layer 7 awareness.

- EVENT-BASED TRIGGERS that automatically change policy in response to specified real-time network traffic conditions by dynamically enabling traffic shaping or filtering.

Yep. Just the device to place in the link between Vodafone Egypt and the rest of the Internet when you want to be able to mess with smartphone users. Citizen Lab also noted that they're unable to conclude whether this device sits on the Telecom Egypt side or the Vodafone Egypt side of the link. But they suspect it's within Vodafone Egypt's network, because precisely targeting injections at an individual specific Vodafone subscriber would require integration with Vodafone's subscriber database. So that's also part of the package. And given that the injection is operating inside Egypt, that this class of spyware is sold to government agencies, and that Egypt is a known Predator customer, Citizen Lab believes that it's highly unlikely that this could have been occurring outside of the purview of Egyptian authorities.



And as if all of this wasn't enough, Ahmed additionally received three SMS messages as far back as September 2021, as well as May and September of this year. All three posed as messages originating from WhatsApp. The fraudulent messages invited him to visit a link contained within the message to "terminate" what the messages claimed was a new login to his WhatsApp account.

Approximately 2 minutes and 30 seconds after Ahmed read the first September 15th 2021 message, the Predator spyware **was** installed on his phone. The researchers suspect that he clicked the message's link and triggered the installation. Since the 2023 messages contain nearly identical bait content, they believe that those messages were also attempts to install the Predator spyware on Ahmed's phone.

And, as if the point hasn't been driven home enough, Citizen Lab's write-up of their research concluded:

> *Our report reveals the potential insecurities that run through the entire spectrum of the telecommunications ecosystem, including at the network layer, which can be exploited to inject malware on an unsuspecting users' device. Our internet communications are routed through many networks and middleboxes, some of which can be mis-used for malicious purposes, particularly if network requests flowing through them are not protected with cryptography. Although great strides have been made in recent years to "encrypt the web", users still occasionally visit websites without HTTPS, and a single non-HTTPS website visit can result in spyware infection. This report should serve as a reminder of the importance of achieving a 100% rate of HTTPS adoption.*

Just for the record, I never suggested that exactly this wasn't possible. In fact, I outlined exactly this architecture for such attacks. But there's also no question that HTTPS would have robustly blocked this particular attack. Not the SMS variant, but certainly the HTTP attack which relied upon on-the-fly traffic interception. Unfortunately, the only way to really prevent such attacks would be to remove HTTP from the Internet and that still seems extreme. The other counter-measure would be to display very scary warning messages. But as we know, users typically just push past anything that gets in their way, so that would be less effective.

Another idea might be to begin neutering what webpages delivered over HTTP can do. Like how about disabling all scripting on HTTP sites. They would still load but they would be restricted to the sorts of old school content that such sites often have while potentially dangerous and abuse prone capabilities would be absent. And this might incentivise their owners to step up to HTTPS which, after all, is no longer difficult or expensive.