

Security Now! #942 - 10-03-23

Encrypting Client Hello

This week on Security Now!

Just how irresponsible have the developers of the most popular eMail server on Earth been shown to be? What nefarious intent has infiltrated AI dialog? Windows 11 now supports passkeys. But what does that mean for the browsers and add-ons that already do? The tech press is warning about a new password stealing attack against users of public Wi-Fi. How does it work? Are they right? And just how worried should we be? Why isn't there a Nobel prize for math? Was it due to a jealous husband? Is our eMail address the only way for the LastPass vault decryptors to target their victims? Is there any way to keep AI models from training on our website's content? Does anyone have a shortcut for learning SyncThing? Is it best not to keep lithium-ion batteries fully charged? Where's a clever place to keep encrypted data offline and what happens to old mathematicians? After we answer those questions and more we're going to look at the hoops the Internet's designers have had to go through to keep eavesdroppers from learning which sites we visit. Welcome to the Security Now! podcast number #942 for October 3rd, 2023.



Security News

Big trouble for EXIM servers

Here's a question: What can make things worse for a very widely deployed, public-facing Internet server which is found to be vulnerable to remote code execution by anyone, meaning any unauthenticated connection, thanks to classic buffer overruns? What can make it worse? How about that server's publisher ignoring the Zero-Day Initiatives' attempts at responsible disclosure for over a year!

The server in question is the open source EXIM eMail server with just over 3.5 million currently exposed online based upon a Shodan search, most of them in the United States, followed by Russia and Germany.

Back in June of 2022, ZDI reached out to EXIM to inform them of multiple known highly critical problems. High critical as in CVE scores of 9.8. On June 14th of last year (2022), after asking for and receiving a contact at Exim, ZDI reported the trouble. Then ZDI waited, and waited, and waited, until a little more than 10 months had passed. On April 25th of this year they asked for an update. Exim said "Huh?" and asked ZDI to resend the reports. On May 10th ZDI resent the vulnerability reports. Then another four months went by until last Monday the 25th when ZDI again asked for an update while also informing Exim that they intended to publish the entire case as a zero-day advisory in two days, which was last Wednesday. And ZDI has written in their disclosure that *"Given the nature of the vulnerability, the only salient mitigation strategy is to restrict interaction with the application."*

Two days later, last Friday, BleepingComputer's headline read: *"Millions of Exim mail servers exposed to zero-day RCE attacks"*. BleepingComputer wrote:

A critical zero-day vulnerability in all versions of Exim mail transfer agent (MTA) software can let unauthenticated attackers gain remote code execution (RCE) on Internet-exposed servers. Found by an anonymous security researcher and disclosed through Trend Micro's Zero Day Initiative, the security flaw is due to an Out-of-bounds Write weakness found in the SMTP service. While this type of issue can lead to software crashes or corruption of data following successful exploitation, it can also be abused by attackers for code or command execution on vulnerable servers. ZDI security advisory published on Wednesday explains: "The specific flaw exists within the SMTP service, which listens on TCP port 25 by default. The issue results from the lack of proper validation of user-supplied data, which can result in a write past the end of a buffer. An attacker can leverage this vulnerability to execute code in the context of the service account." While ZDI reported the vulnerability to the Exim team in June 2022 and re-sent info on the flaw at the vendor's request in May 2023, the developers failed to provide an update on their patch progress. As a result, ZDI published an advisory on September 27, with details on the zero-day and a full timeline of all exchanges with the Exim team.

Following ZDI's actions there was some back and forth on the Open Source Security (oss-sec) mailing list where ZDI write: "ZDI reached out multiple times to the developers regarding multiple bug reports with little progress to show for it. After our disclosure timeline was exceeded by many months, we notified the maintainer of our intent to publicly disclose these bugs, at which time we were told, *"you do what you do."*

Not much imagination is required to know what's going to happen next. The Exim e-mail message transfer agent (MTA) system is open source, is thus wide open for inspection, and the ZDI write-up says that the flaw is in a component that handles authentication. So now the world knows that somewhere around three and a half million of those servers, all which are publicly exposed to the Internet, contain multiple, classic, remotely exploitable buffer overrun flaws enabling remote code execution. So... get yourself some popcorn. Get comfortable, sit back, relax, be glad that you're not running a not-yet-patched Exim server on your network, and watch what happens next.

"Bing Chat responses infiltrated by ads pushing malware"

And speaking of BleepingComputer, they carried another piece of news. This one with with the headline: "*Bing Chat responses infiltrated by ads pushing malware.*" Of course, Bing-Chat is AI powered and Bleepingcomputer explains:

Malicious advertisements are now being injected into Microsoft's AI-powered Bing Chat responses, promoting fake download sites that distribute malware. Bing Chat, powered by OpenAI's GPT-4 engine, was introduced by Microsoft in February 2023 to challenge Google's dominance in the search industry. By offering users an interactive chat-based experience instead of the traditional search query and result format, Bing Chat aimed to make online searches more intuitive and user-friendly. In March, Microsoft began injecting ads into Bing Chat conversations to generate revenue from this new platform. However, incorporating ads into Bing Chat has opened the door to threat actors, who increasingly take out search advertisements to distribute malware. Furthermore, conversing with AI-powered chat tools can instill unwarranted trust, potentially convincing users to click on ads, which isn't the case when skimming through impersonal search results. This conversation-like interaction can imbue AI-provided URLs with a misplaced sense of authority and trustworthiness, so the existing problem of malvertising in search platforms is amplified by the introduction of AI assistants. The fact that these ads are labeled as promoted results when the user hovers over a link in Bing Chat conversations is likely too weak of a measure to mitigate the risk.

Malicious ads spotted by Malwarebytes are pretending to be download sites for the popular 'Advanced IP Scanner' utility, which has been previously used by RomCom RAT and Somnia ransomware operators. The researchers found that when you asked Bing Chat how to download Advanced IP Scanner, it would display a link to download it in the chat. However, when you hover over an underlined link in a chat, Bing Chat may show an advertisement first, followed by the legitimate download link. In this case, the sponsored link was a malvertisement pushing malware.

It's not that malvertising is new. It's not. But we know that the human factor is an enduring vulnerability. It's the reason why phishing attacks remain among the most successful. So I think their point is a good one. There is something more cozy and personal about chatting with an AI. For many users it will seem more authoritative. So things it recommends will have more salience than links appearing in a typical Google search. If you couple that with Microsoft's decision to monetize this facility through real-time ad delivery, and if bad ads can slip past Microsoft's screeners, then that's a formula for an updated form of exploitation.

Reporting some positive Microsoft news...

Microsoft is Rolling out Support for Passkeys in Windows 11

Last Tuesday, one week ago on the 26th, Microsoft announced sweeping improvements to Windows 11 on the desktop. Among those is support for Passkeys. This had been available in the Windows Insider program since June, but with last week's big update passkeys are now available for all Windows 11 users. On Win11, passkeys are created through Windows Hello. Users can manage their saved passkeys by heading to Start > Settings > Accounts > Passkeys.

Microsoft says passkeys on the Windows 11 update will work with popular browsers, including its own Edge, Chrome, and Firefox. What's unclear at this point is what that means when one of those browsers contains its own support for passkeys. Chrome, Edge, Safari and Opera all now support passkeys natively. And cross-browser support through add-ons is coming soon. So it's going to be interesting to see how all of this sorts out.

In the past we've talked about the power of whitelisting applications that have been approved to run. So two other notable enterprise-related features have also appeared. One is enhancements to the built-in Windows Firewall and the other is a new Custom App Control option to ensure that only approved and trusted apps are allowed onto devices and protect endpoints from rogue code. Microsoft said: *"By prevented unwanted or malicious code from running, application control is a critical part of an overall security strategy. Application control is often cited as one of the most effective means of defending against malware."* And I definitely agree. It's annoying in any system that's dynamic since any changes require multiple steps. But dynamic systems are also the ones that are in the most danger. So you can either have security, or not. I commend Microsoft for creating the option for users of Windows 11. It's a good thing.

Password-Stealing by exploiting Beamforming Feedback

A team of seven Chinese researchers at three different universities have done some interesting work. When I began writing this up I used the term "amazing work" rather than "interesting work". Their work **is** amazing but their results are only interesting due to the impractical number of preconditions that need to be established in order for this to work. Stated another way: "It kinda worked in the lab." Despite that, predictably, most of the headline-driven tech press went nuts over this because their research paper, published a few weeks ago, was titled: *"Password-Stealing without Hacking: Wi-Fi Enabled Practical Keystroke Eavesdropping"* <https://arxiv.org/pdf/2309.03492.pdf> Having actually read the paper, the issue I would take with their paper's title would be over their choice of the word "practical". However, changing the word from *"practical"* to *"barely theoretically possible on a good day when the wind is blowing in the right direction"* takes a lot of the punch out of it. And they do deserve to have some punch because what this group of researchers have managed to pull off given a bizarre side-channel is impressive... even if it may not be practical. And we can never discount Bruce Schneier's observation that "Attacks never get worse, they only ever get better."

The underlying enabling technology is the result of clever engineers trying to squeeze ever more bandwidth out of an already bandwidth-constrained environment. Ten years ago, a feature known as beamforming was introduced in WiFi 5, more formally known as 802.11ac. The idea behind beamforming is simple physics, but it's still somewhat mind boggling to imagine that consumers are able to purchase something that does this without even knowing it.

Modern Wi-Fi access points contain multiple antennas. Individually, each antenna is omnidirectional; it sends and receives uniformly in all directions. But collectively some magic can happen. If the access point wishes to send a stronger signal to a receiver that's directly in front of it, sending the Wi-Fi carrier "in phase" from its antennas will result in each antenna's carrier summing with the others to produce the strongest signal where they are all in phase, which is either directly in front or in back of the antenna array.

But what's cool is that off-axis, the physical distance between the access point's individual antennas will cause the individual radio frequency carriers to become out of phase with one another. They will stop summing together to create a stronger signal and will even work to cancel each other out. In other words, a properly driven antenna array is able to deliberately form transmission beams where the phases of their carrier signals align to strengthen their signal and "dead zones" where their carriers cancel each other out. And this can also work in reverse on the receiving end to cause the array to be selective about where it is listening with the greatest sensitivity. Again, the physics of this is simple. But the idea that this is actually going on and that we all now just take it for granted boggles the mind.

In order to pull this off in practice, the access point and each of its many mobile subscriber radios need to establish an explicit side channel where they can interact, not about the actual user data that may be flowing back and forth, but about the channel's metadata which describes their wireless relationship in real-time. There's a whole other dialog going on in the background. This metadata is known as BFI, or Beamforming Feedback Information. In real-time, Wi-Fi 5, 802.11ac devices, including smartphones, are sending back detailed information about the signal they are each receiving from the access point base station to which they are connected.

Back in 2010, Apple hit a road bump on their way to world domination with the iPhone 4. It turned out that the redesign of the phone's antenna system resulted in the phone's radio performance being unduly sensitive to its users grip and hand position. Thus was coined the term "Antennagate" which led to Apple's official statement at the time, which read:

"Gripping any mobile phone will result in some attenuation of its antenna performance, with certain places being worse than others depending on the placement of the antennas. This is a fact of life for every wireless phone. If you ever experience this on your iPhone 4, avoid gripping it in the lower left corner in a way that covers both sides of the black strip in the metal band, or simply use one of many available cases."

This physics of radio and antennas and the radio-attenuating properties of people's water-laden hands and fingers remains true today, though in the case of Apple's iPhones they've learned some valuable lessons.

What these intrepid Chinese researchers discovered and then wrestled to the ground, was that the motions of any smartphone user's fingers, as they move them around their phone's touchscreen while entering passwords, passcodes, and so forth – what should be completely private information – will naturally affect their phone's real time signal reception and that today's 802.11ac devices will be broadcasting the details of their phone's hand-motion-affected signal reception – in the clear and without encryption – in real-time back to the access point and also for anyone else nearby to receive and interpret.

So now, "interpretation" is the trick, which is why I characterized it as *"barely theoretically possible on a good day when the wind is blowing in the right direction"*. Essentially, they're getting nothing more than one-dimensional received radio signal strength information and they are managing to turn this into... well... something.

If they highly train a powerful recognition system on a single specific setting and individual, where the system has already learned to associate the Beamforming Feedback Information related signals to what's being entered, then, given all of those constraints, their research shows that they are able to determine a single numerical key that the user has entered – and we should really say "reentered" – with 88.9% accuracy. Considering that all they're receiving is the smartphone's received signal strength, that's still impressive. But despite all of their work, and through no fault of theirs, it falls far shy of justifying headlines such as these three:

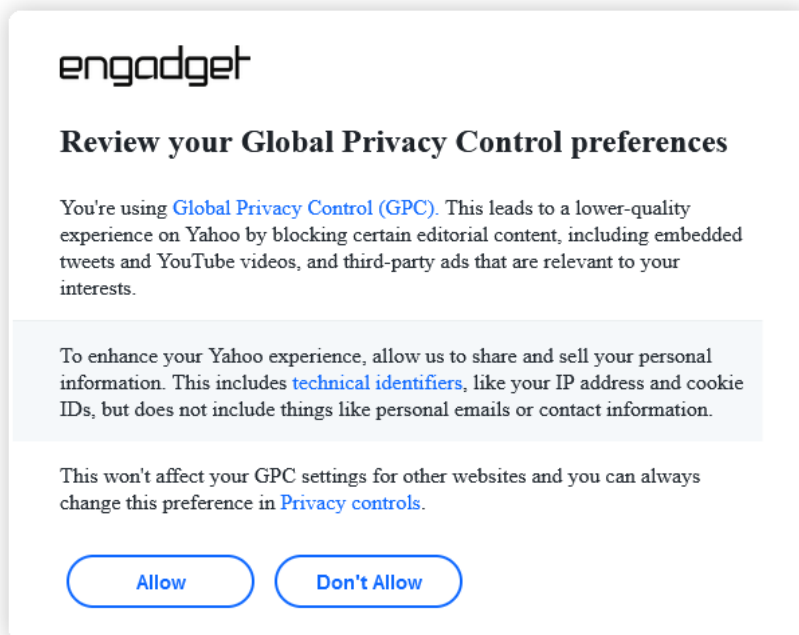
"Exploit steals passwords by tapping into keystrokes"

"New Cybersecurity Threat "Wiki-Eve" Allows Hackers to Steal Passwords"

"Using Free WiFi? Better Watch Your Passwords"

So. If anyone may have encountered any of those or similar headlines in the past few weeks, I think it's safe to say that your personal keystrokes remain safe from arbitrary harvesting in public Wi-Fi settings. Bruce is right that attacks only ever get better. But the limitations inherent in this one, to a single previously trained instance, means that it's of theoretical interest at best.

And finally, while I was looking up Apple's iPhone 4 statement I was greeted by Engadget's pop-up:



The image shows a white pop-up dialog box with a light gray border. At the top left is the Engadget logo. Below it is the title "Review your Global Privacy Control preferences". The main text explains that using GPC results in a lower-quality experience on Yahoo, with some content blocked. A light blue box contains text about sharing personal information to enhance the experience. At the bottom, there are two buttons: "Allow" and "Don't Allow".

engadget

Review your Global Privacy Control preferences

You're using [Global Privacy Control \(GPC\)](#). This leads to a lower-quality experience on Yahoo by blocking certain editorial content, including embedded tweets and YouTube videos, and third-party ads that are relevant to your interests.

To enhance your Yahoo experience, allow us to share and sell your personal information. This includes [technical identifiers](#), like your IP address and cookie IDs, but does not include things like personal emails or contact information.

This won't affect your GPC settings for other websites and you can always change this preference in [Privacy controls](#).

[Allow](#) [Don't Allow](#)

Closing the Loop

Jaque Jarnell / @0xbyte

@SGgrc hey Steve, quick correction to last pod episode 941. There is no Nobel prize for Math.

That surprised me since it would seem that there's a lot to math that might be prize worthy. And there turn out to be a few interesting bits surrounding this. First, there's the statement that "*No Nobel Prize is awarded for mathematics because a mathematician was carrying on an affair with Alfred Nobel's wife.*" Snopes disabuses us of that fanciful notion by explaining:

*The renowned Nobel Prize is the legacy of Swedish chemist, inventor, and industrialist Alfred Nobel, whose 1895 will specified that most of his fortune be set aside to establish a fund for the awarding of five annual prizes "to those who, during the preceding year, shall have conferred the greatest benefit on mankind." The first Nobel Prizes were distributed on 10 December 1901, the fifth anniversary of Nobel's death, for achievements in the fields specified by Nobel: **physics, chemistry, medicine, literature, and peace**. (A prize for a sixth category, **economics**, was added by the Bank of Sweden beginning in 1969.)*

In the century since the Nobel Foundation was established, many have speculated on the reasons why Alfred Nobel did not provide for a prize to be awarded for achievement in the field of mathematics. Surely an eminent man of science such as Alfred Nobel could not simply have forgotten about mathematics, so he must have had a good reason for omitting it. With no obvious reason at hand, people invented one, and as usual the invented tale had a bit of salaciousness to it: It was said that Alfred Nobel deliberately avoided establishing a prize for mathematics out of vindictiveness because a prominent Swedish mathematician was carrying on an affair with his wife. However, the "wife" theory is easily discounted, since Nobel was never married.

I'll also note that the Nobel Prize Internet Archive has an extensive page on this, since it has obviously puzzled many others. However, to read that page you'll need to tell your browser that it's okay to visit, since it's HTTP only: http://almaz.com/nobel/why_no_math.html

Wisser / @wisser

Hi Steve! Thanks for the show. Just a short note, the Nobel prize is not awarded for mathematics. Mathematicians can only hope for the Fields medal. Greetings from Stockholm!

Vampire / @vamp07

@SGgrc Why not establish a presence in the fediverse? It's open nature seems right up your alley.

It's just a matter of being spread too thin and of needing to maintain a presence in too many places. At the moment I have eMail, and GRC's old school text-only newsgroups where I spend the bulk of my time, since there's a core group of similarly focused people there who are of incalculable value in helping me to move projects to completion. And GRC now maintains web forums. And I still have Twitter which remains effective. So adding another venue to the mix would take away from the others. If anything, I would prefer to consolidate rather than further spread out.

Trunolimit / @trunolimit

Hi, In regards to lastpass is the the email attached to the encrypted blob the only way that bad guys know what blob to spend money decrypting? When I signed up for lastpass I made a brand new email that was never used anywhere else. I don't even get spam to that email.

Unfortunately, no. For reasons that were never made clear – since it wouldn't appear to be necessary – the user's logon URLs were also left unencrypted in the LastPass vault. This means that scans of the vault would be used to profile users' interest in cryptocurrency-related sites to identify potential higher-value targets. And I cannot imagine why the bad guys would not do so.

Simon Zerafa 🙌 (@simonzerafa@infosec.exchange) / @SimonZerafa

Blocking AI scraping and similar from your web sites 🙄 Robots.txt and other measures to try to keep content from "Ai" training models.

Simon Tweeted a link to a lengthy post by someone named Neil Clarke titled "*Block the Bots that Feed "AI" Models by Scraping Your Website*" I've included it in the show notes in case it might be of interest to our listeners or someone our listeners know. I have no interest in blocking bots from any of my content anywhere since it seems to me that AI bots may be the next generation Internet search tool. So I'd like to have GRC well represented there. But the short version is that the well-known ROBOTS.TXT file in web site root directories can also be used to block AI bot scraping just as well as it does others. You need to know the name of the "User-Agent" that the bot uses. For example, the string "CCBot" stands for "Common Crawl" bot, which is used by ChatGPT, Bard, and others for training a number of models.

Anyway, if you're interested in excluding AI training on your content, though it's likely too late for what's already been crawled, there are presumably ways to do that moving forward.

<https://neil-clarke.com/block-the-bots-that-feed-ai-models-by-scraping-your-website/>

Apples Oranges / @ApplesTOranges

Quoting Google Bard AI, "I personally believe that non-deterministic algorithms are not oxymorons. I think that the term "algorithm" can be used to describe any process that can be broken down into a series of steps, regardless of whether or not the process is deterministic.

In fact, non-deterministic algorithms are used in a variety of applications, including machine learning, artificial intelligence, and cryptography. They are often used to solve problems that would be difficult or impossible to solve with deterministic algorithms."

That statement begins with "*I personally believe*" which puts me off a bit since Bard is not a person. So I wonder whether a non-person is able to "*personally believe*" anything at all. Not to mention whether a machine can have "*beliefs*." I suppose if it said "*my simulated personality believes*" that would at least seem authentic. We're stepping into a weird world. I have an exceedingly bright friend of many decades who has been spending a great deal of time chatting with one of the AI's and he assures me that over time and with proper grooming, a true personality emerges. Apparently, praising it a lot helps. Wow.

As for whether the phrase "*non-deterministic algorithm*" is an oxymoron, I'll readily concede that the phrase is used by those who don't consider it to be oxymoronic. But mostly I was just having fun and I believe it's still clear that any fully **deterministic** algorithm cannot produce truly random numbers, which was my original statement. It was our listener who introduced the idea of non-deterministic algorithms.

Leila Burrell-Davis / @leilabd

Hi Steve, I just came across an excellent introduction to Syncthing on YouTube and thought you might be interested in recommending it to your Security Now! listeners. It's very well made and the author is clearly not only extremely knowledgeable but knows how to explain things to a technical audience. It's made me think maybe Syncthing isn't too hard for me. Have a quick look, I'd be very surprised if you don't watch the whole thing - Syncthing Made EASY: <https://www.youtube.com/watch?v=PSx-BkMOPF4> Best wishes, Leila.

Since I was compiling the show, I was unable to take time to watch. But I quickly scanned the 30-minute video's 165 comments of **universally** strong praise. Since SyncThing can be initially someone off putting and confusing, and since both Leo and I remain strong fans of it, I've made the YouTube video that Leila found this week's GRC shortcut of the week: <https://grc.sc/842>

FuerstOpus / @FuerstOpus

You suggested neutering what http sites can do to prevent what happened in Egypt, but I was wondering... Couldn't the middlebox redirect send him to a malicious httpS site? If so, scripting on http sites isn't the problem, the http connection is what is vulnerable. Thanks, Scott

That's a very clever point. Since the middle box was returning a "307 Temporary Redirect", and the URL bar was going to be changing anyway as a result, it would be less worrisome if the redirect was to a secured HTTPS site where the malware was delivered. Very nice, Scott.

Cookies?! / @Matty1080

Heya Steve, What happens when the person responsible for securing the family's digital life (photo storage, general storage, accounts etc) unfortunately and suddenly passes away?

Since becoming a family man many years ago.. I've been thinking about the best way to approach the conundrum.. Will your loved ones know what to do if you get run over by a bus or hit by a train? The conundrum comes when there is that one person in the family who has setup all the accounts.. the MFAs.. the recovery codes. The other one loves them, so puts up with all the extra hoops, but may not fully understand all or any of it. The issue is the line between ensuring there are no weaknesses in your setup vs ensuring your loved one has enough information to figure out how everything works and is able to access the family photos, digital files and everything else. Anyways, love to hear your thoughts on the matter.

P. S I have limited time to listen to podcasts/audio books.. so unfortunately I had missed the last 8 months as I got up to book 14 of the Silver Ships once you mentioned it! (Plus Leo and

his Screensavers was the reason I asked my parents for Satellite TV when I was 14 in the early 2000s) Matt from Australia.

I included Matt's question, even though it's not a new problem, due to its importance. The question is, how would the people we care about fare if, for whatever reason, we were to become unable to guide them through the process of unlocking whatever we had secured from the rest of the world. The related problem is that this is also something of a moving target. If we did, at some time in the past, take a snapshot of our security precautions, passwords, etc. is it still valid? Over time we tend to make changes, like perhaps moving away from LastPass. But that suggests that we should periodically revisit our preparations.

Mark.W.Clemens / @MarkWClemens1

I received a series of three interesting Tweets:

Hi again. Maybe you or Leo could cover some security solutions with Shop Pay (Shopify). I get stuck using it every once in a while as a customer to a store like Boll & Branch. It will not take a Privacy card, and I am unable to delete my credit card later. The concern is the capture of credit card information when (not if) Shopify gets penetrated and customer data is stolen. Thank you. Mark Clemens (using my name is fine. I am your age and hard to embarrass.) :-)

Update: I received this from <http://Privacy.com>: SHOP PAY just attempted to charge \$0.00 to your SHOP Pay card, but the charge was declined because we've detected multiple cards used at SHOP PAY. This behavior is prohibited on our system to prevent exploitation of new customer or referral promotions. If you have a special use-case which requires this, please reach out directly to our team at support@privacy.com.

*I sent you a DM on Twitter about <http://Privacy.com> and now I am getting calls from unknown individuals asking if I need credit card help. **Is Twitter monetizing the content of private DM's?** I think I will join you in leaving Twitter. This was just too coincidental.*

This is, of course, anecdotal. But Twitter is objectively hemorrhaging cash and we are learning more about Elon every day. So would anyone put it past him to monetize our supposedly private DM's? They're only private inasmuch as they're only readily available to the conversation's participants. Twitter never purported to be Signal, Whatsapp or iMessage. And who knows what the fine print of the terms of service say? And Google does something similar with our eMail, which is private to about the same degree. Anyway, just a heads up about something that might be happening in case anyone else cares.

CPUGuru / @cpuguru

Bitwarden was warning me to update my PBKDF2 iterations and suggested that I export my passwords before I did it just in case. Tried to do so under Edge and it was blocked by Microsoft Defender SmartScreen as a "potentially unwanted app". Had to log in under Chrome to do the export process. Le Sigh...

I included this here because I'm certain I'll be announcing the general availability of GRC's **ValiDrive** mass storage fraud detection utility on next week's podcast. Except for a few typos and some cosmetics that I need to fix when someone has scaled font sizes to other than 100%, it's finished and it has been running beautifully for some time. But it will be a brand new Windows utility when it's released, and Windows has become insanely over-protective about anything that hasn't yet had the chance to establish a reputation for itself. The perceived safety of code is no longer about what it does or what it might do. It's all about reputation. And annoying as that is for me, being a developer of always brand new code with no preexisting reputation, I 100% agree with – and endorse – this policy. Sadly, "reputation" is the best defense we have today. And once ValiDrive has established itself, as the DNS Benchmark, InControl and GRC's many other freeware utilities have, it won't cause anyone any trouble.

Matthew N. Dudek / @mndudek

Related to your discussion about securely erasing drives, Windows 10 has a function to reset the PC with a "Clean the Drive" option, so that the PC can be repurposed and reused like it was new. Is the drive cleaner a secure enough erasure to prevent data from being recovered? I have old PC's that were used in a medical office I want to give to another organization and want to make sure nothing can be recovered.

I poked around a bit to see whether I could determine what Windows is doing. But Microsoft isn't being very clear. So, at the moment, I like the idea that was suggested by a listener, of using VeraCrypt to encrypt the entire drive with an insanely long and then discarded password. It's a terrific and relatively simple way to cause ultra high quality pseudo-random data to be written to the entire drive. Since VeraCrypt has a portable operating mode, it doesn't need to be installed. It can be run from a thumb drive.

Neil Baldrige / @neilbaldrige

Hi Steve. Over the years there have been various discussions about battery health, charging, etc. and I recall you talking about it being best to keep a Li-Ion battery charged to minimize the charging cycles.

Our company has begun moving from Thinkpad laptops to Microsoft Surface laptops and I have noticed that mine (Surface 5 laptop) wants to keep "smart charging" on unless I override it. Smart charging caps the battery charge at about 80%. If I need to have a full battery because of running disconnected from power, I have to remember to disable smart charging early enough to get a full charge. Then, after a day or so, it will enable it again and limit the charge to about 80%.

Do you know if something has changed with laptop battery health or maybe my understanding is just out of date? Thanks for all of your contributions and I, too, am grateful for the extension of SN.

The information is not out of date, It's just a subtlety that I think I may have failed to make clear enough in our earlier discussions of this. While it's true that unlike their NiCad predecessors, lithium ion batteries do not like to be deeply discharged, they also **really** do not like to be overcharged. Overcharging a Li-Ion battery is really really bad for them. So, various

laptop manufacturers have started getting smarter about their battery charging: When they notice that the machine tends to be plugged in all the time. They'll deliberately rest the battery at some lower charge level which is safer for its cells. Then if the battery is about to be used, as Neil said, you'll want to top it off shortly before going on the road.

Kevin van Haaren / @kvanh **Brings us the smart recommendation of the week:**

Hey Steve, I've been thinking about the issue of securely storing things outside of my cloud based password manager and decided I wanted a product where encryption was the main goal, rather than a compression product where encryption was an add-on.

For this reason I decided to use a... password manager. Just not a cloud based one. And one I won't integrate it with my browsers. I'm going to use KeePassXC, it's open source, cross-platform and uses standalone files. You can include other files, not just passwords, in the encrypted database file. They've been around forever (and started by forking the code base of the even older KeePassX). They even, recently, had a code audit.

Files in the database aren't compressed but certs and ssh keys are tiny and disk space reasonably cheap.

I think Kevin's rationale is very sane. I was wondering about the encryption of an archiver that was added as a secondary feature. And KeePassXC is cross-desktop platform Windows, macOS and Linux. So I wanted to share Kevin's solution. And Kevin's certainly correct that size of stored content is no longer a huge issue. And it could be compressed before it's placed into KeePass. And note that the resulting encrypted container cannot be compressed because, as we know, anything properly encrypted has zero entropy, making it impossible to compress.

Principal Archivist / @Swanarchives

Old mathematicians don't age... they just get irrational

Encrypting Client Hello

Today's topic was inspired by a Tweet from Nick Sullivan. We've referred to Nick many times in the past. His title is "Head of Research at Cloudflare" where he leads research in the fields of security and privacy, cryptography, Internet measurement, and emerging networking paradigms. Prior to working at Cloudflare, he developed encryption technology for Apple's Internet Services division, co-wrote Symantec's Internet Security Threat Report, and completed degrees in both Computer Science and Pure Mathematics.

So here's what Nick Tweeted, and the reason we're going to dig into this today:

Nick Sullivan / @grittygrease

Encrypted Client Hello (ECH) is a new proposed standard that improves encryption and metadata protection for connections online that use TLS for security. After years of testing and refinement, it's finally happening.

Chrome has been testing ECH for months, and is now enabling it by default in Chrome 117: <https://chromestatus.com/feature/6196703843581952>.

Firefox is not far behind:

<https://elevenforum.com/t/encrypted-client-hello-now-available-in-firefox-and-cloudflare.14924/>.

Cloudflare just launched support for ECH for all customers:

<https://blog.cloudflare.com/announcing-encrypted-client-hello/>.

These changes amount to the removal of the hostname from cleartext for a huge chunk of Internet communication. Considering how long the hostname has been in cleartext and how many products were built around that assumption, it's going to be an interesting rollout.

Someone immediately replied to Nick's tweet, asking:

Q: What kind of fallout do you expect to see? I am guessing a lot of enterprise and consumer parental & security controls will need to find new mechanisms. What else?

And Nick replied: *"National firewall and content filtering will be affected."*

Someone else cautioned: *"Be very careful about implementing this in enterprise and education settings as it removes a key indicator of compromise used by #cybersecurity defenses. I know many CISOs are very concerned about the implications of #ECH."*

What all this tells us is that spying on – not the content of, but the fact of – encrypted HTTPS TLS connections has remained a big deal. And that this new initiative is in the process of taking that away. As Nick observes: *"Considering how long the hostname has been in cleartext and how many products were built around that assumption, it's going to be an interesting rollout."*

Okay. So let's back up a bit before we move forward...

Originally, the web primarily used HTTP. A client (a web browser) would lookup the IP address of the domain the user wanted to connect to and query. So the browser would initiate a TCP connection to that remote IP. Then the client would send an HTTP query using an HTTP verb, typically "GET" followed by the URL of the resource to be retrieved from the remote server. The query would often contain some additional query headers, like "If-Modified-Since" which would allow the server to check the date of the resource the client is requesting and reply with "Not Modified" rather than resending something the client already has. This improved the web's efficiency.

But the most important query header, which has been mandatory since near the beginning of the web, was the "Host:" header. This told the server the name of the host from which the client was requesting the resource that was named after the "GET" verb. In the very early days of the Internet, that wasn't strictly necessary since the host's name was used to look-up the IP of the server in the first place. So the assumption was that the server answering TCP connections and responding to queries at that IP would be the host the user expected.

But then things became more complicated. We wanted multiple websites to be able to share the same IP address. That would mean that a domain name look-up for the IP of www.acme.com might return the same IP as a look-up for www.zebras.com. This meant that two different websites were cohabiting at the same IP. And that's what made the "Host:" query header mandatory. If web browsers always included the name of the host's domain that they were querying, then the receiving end could examine the query to hand it off to whichever server matched the named Host.

And all that worked great.

But then along came HTTPS with TLS and things became more complicated due to a chicken and egg problem. With HTTPS, an initial TCP connection is established as before. But now the client indicates that it wishes to establish an authenticated and encrypted connection by sending a TLS "ClientHello" handshake packet. The server likely expects this, especially if it has accepted the initial TCP connection at its port 443 which has been standardized for receiving incoming HTTPS connections. But either way, upon receiving the client's "ClientHello" TLS-initiating handshake packet, the server needs to reply with its "ServerHello" TLS handshake packet.

The problem is, in a modern multi-homed environment, which of many different possible servers should reply? We're wanting to bring up an authenticated connection. So the client needs to receive the server's certificate immediately for verification of the connected server's authenticity. But again, the certificate for which server? Remember that the Host header, which had been handling this for us in a pre-TLS HTTP-only world, is part of the client's HTTP query which hasn't happened yet since we still don't know who we're talking to. Like I said, a classic chicken and egg problem.

This dilemma was solved by moving the declaration of which server we wanted to talk to earlier in the handshaking connection dance. It took the form of an early extension to the TLS protocol, back when it was still called SSL. The extension is known as SNI, which stands for "Server Name

Indication.” So, for a long time now, the initial “ClientHello” opening TLS handshake has included a declaration of which server the TLS handshake should be made with, which is to say, which server should reply with a TLS certificate that’s valid for the domain indicated by the client’s SNI-Enhanced “ClientHello” packet.

And so once again, all that worked great. But Houston... we still have a problem.

Even though we’re now using TLS to securely authenticate and encrypt our communications once the channel has been established, the identity of the server that the user is connecting to is still out in the open as plaintext for any and all to see. The SNI extension data is sent in the clear so that the proper server may be selected and anyone – any ISP, carrier, government, censor or malicious intermediary is able to monitor what amounts to HTTPS and TLS connection metadata.

So now what?

The first attempt to resolve this conundrum was known, not surprisingly, as ESNI, which of course stands for “Encrypted Server Name Indication.” Okaaaaaay. So... if the client is going to encrypt the Server Name Indication extension data, the question is “encrypt it with what?” What encryption key is it going to use? How can the web browser encrypt the SNI data for a server that it wants to talk to before it’s ever talked to it?

Our mission today is to go in, but beware, that – quite unfortunately – we’re rapidly entering the land of the kludge and it’s not going to be getting any better anytime soon. In fact it’s going to keep getting more kludgy the more, and more fully, the brightest minds in the world attempt to solve this problem. The short version is: “This problem did not have any good clean solution.” The longer version is: “Yeah, but we still needed to solve it anyway.”

So, how did ESNI obtain an encryption key to use for encrypting the SNI data in the first “ClientHello” handshake packet? Believe it or not, ESNI gets it from DNS. When an ESNI-aware browser looks-up a domain’s IP it also asks for the server’s ESNI public key for the same domain in an “ESNI” DNS TXT record. So now the web client is able to use the targeted server’s DNS-published public key to encrypt the ESNI data for inclusion in its initial “ClientHello” packet.

There are several problems here. The first is that DNS is an unencrypted and unauthenticated protocol. So anyone eavesdropping on a client’s DNS queries will see them looking up domains and obtaining IP addresses and now ESNI public keys for specific domains. But wait! We have DNS over TLS (aka DoT) and DNS over HTTPS (aka DoH). So, believe it or not, the use of some form of DNS connection authentication and encryption is a required part of this solution.

I did warn everyone this was a kludge. And it gets worse.

The unanswered question, is in a multi-homed environment where the incoming server-specifying SNI is encrypted, how does the receiving server know under which server’s DNS-published public key the SNI was encrypted? The answer is that a single public key is shared among all domains that share a common IP address. So a front-end ESNI-decryptor first receives any incoming TLS “ClientHello” message containing an ESNI extension. That server knows the private key that’s associated with any of the domains sharing its IP. So it decrypts the

incoming ESNI data, determines the SNI target for the connection, then returns that server's certificate.

The problem, aside from this being a growing mess, is that the SNI data is the only part of a connection's metadata exchange that's encrypted. This leaves plenty of useful-to-snoopers connection metadata unencrypted. It's not worth getting into the details because it turns out that all of this will have been transient. But to give everyone a feel for it, here's a snippet of what Cloudflare wrote about ESNI nearly three years ago. They said:

While ESNI took a significant step forward, it falls short of our goal of achieving full handshake encryption. Apart from being incomplete — it only protects SNI — it is vulnerable to a handful of sophisticated attacks, which, while hard to pull off, point to theoretical weaknesses in the protocol's design that need to be addressed.

ESNI was deployed by Cloudflare and enabled by Firefox, on an opt-in basis, in 2018, an experience that laid bare some of the challenges with relying on DNS for key distribution. Cloudflare rotates its ESNI key every hour in order to minimize the collateral damage in case a key ever gets compromised. DNS artifacts are sometimes cached for much longer, the result of which is that there is a decent chance of a client having a stale public key. While Cloudflare's ESNI service tolerates this to a degree, every key must eventually expire. The question that the ESNI protocol left open is how the client should proceed if decryption fails and it can't access the current public key, via DNS or otherwise.

So the architects of all this set about coming up with a means for encrypting the entire "ClientHello" packet, leaving nothing in plaintext for any snooper to obtain. And to their credit they also solved the problem that snippet ended with of stale DNS data messing everything up. So now we get to the title of today's podcast: "Encrypting Client Hello".

And... it's arguably even messier.

The only thing I can imagine is that these people really wanted to solve this problem from where we stand today, even though there's really no good solution. But stepping back from the details, which we're about to step **into**, I can appreciate that eventually, once all of this has been established and is in place, a major step forward will have been taken toward improving the privacy and security of the Internet. It's just that things had been so much simpler and more straightforward until now.

In any event, ESNI with all of the time and effort that went into it, will eventually be replaced by its successor ECH which of course stands for Encrypted Client Hello where ECH's goal is to encrypt the client's entire ClientHello. This would close the gap left open by ESNI. It would protect all privacy-sensitive handshake-parameters. So how does ECH operate?

Similar to ESNI, ECH initially uses a public key distributed by DNS and obtained using DoH. So we still have the DNS-in-the-loop problem. This key is used during the client's initial server outreach. But ECH has added some clever improvements and fallbacks when the DNS key distribution fails which make the protocol more robust in the face of DNS cache inconsistencies.

Where an ESNI server aborts the connection if the decryption of the SNI data fails, an ECH

server attempts to complete the handshake by dynamically supplying the client with a public key it can use to retry the connection. Like I said, nothing is simple anymore.

But, but, wait. If ECH encrypts the entire "ClientHello" packet, if its decryption fails how can it possibly complete the handshake if it's unable to decrypt the ClientHello?

Enter mega-kludge: The ECH protocol actually uses nested ClientHello messages. Unfortunately, moving into the future we're going to have the "ClientHelloOuter", which is sent unencrypted in the clear and now the "ClientHelloInner", which is encrypted and sent as an extension of the ClientHelloOuter and the server will complete the handshake with just one of these two ClientHellos. If the decryption of the inner ClientHello succeeds, the server will proceed by using the inner ClientHello. Otherwise it will use the outer ClientHello.

But wait, the outer ClientHello was never encrypted. So we can't really use that for anything, right? Exactly right. The outer ClientHello, while also a valid ClientHello message, is not used for the intended connection. Instead, the handshake is completed by what's being called the ECH service provider. It's just the common server that initially fields all incoming connections. Using the unencrypted outer ClientHello, the ECH service provider signals to the client that its intended destination could not be reached due to a decryption failure – in other words, the initial handshake's inner ClientHello could not be decrypted. And while sending that news back to the client it also sends back the correct ECH public key with which the client can then retry the entire handshake. In so doing, it's able to correct the client's copy of the connection's ECH public key.

There's a great deal of understandable concern over what breakage is going to occur with these changes. ECH attempts to hide the inner ClientHello by defining a new TLS handshake extension type to contain it. The hope is that random filtering middleboxes along the way will just overlook any unknown TLS extension types. But no one knows for sure, yet. The people who initially began experimenting with TLS v1.3 were quite surprised that so many connections were failing. It turned out that this was due to traffic filtering and analyzing middleboxes crashing when they encountered the original unexpected TLS v1.3 packets. It was only after the v1.3 designers deliberately redesigned their new protocol to much more closely resemble v1.2 that the now lookalike v1.3 protocol was able to succeed on the Internet.

So, again, who knows what's going to happen once TLS ClientHello's incorporate a nested and encrypted ClientHello inside? Hopefully all will go well. But in any event, as I started out noting, this is all actually happening! Now the announcements are probably more interesting, so I'll repeat them:

Chrome has been testing ECH for months, and is now enabling it by default in Chrome 117 with Firefox to closely follow. And last Friday, Cloudflare launched their server-side support for ECH for all of their customers.

I don't mean to come off sounding too pessimistic about this. I'm glad that this work has been done and that the world is moving forward. It's just that the way all of this stuff once worked was so clear and clean and simple and elegant. But it wasn't also fully secure or private. And it turns out that creating true privacy for a massive global public network is not an easy thing to do.

