Encrypting Client Hello

Description: Just how irresponsible have the developers of the most popular email server on Earth been shown to be? What nefarious intent has infiltrated AI dialogue? 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 WiFi. 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?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-942.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-942-lq.mp3

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 #942 for October 3rd, 2023.

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. There's a big flaw in the world's most popular email server, and it's been there for a long time. We'll talk about a new attack on WiFi and passwords that you don't have to worry about. Whew. And why is there no Nobel Prize for Mathematics? Steve Gibson and a whole lot more coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 942, recorded Tuesday, October 3rd, 2023: Encrypting Client Hello.

It's time for Security Now!, the show featuring the star of the TWiT network, Mr. Steve Gibson. And I say that, Steve - hello, Steve.

Steve Gibson: Hello, Leo.

Leo: I saw that because I was in Green Bay, Wisconsin, as you know, last week. Thank you, Ant, for filling in. And Friday we had a meetup. About 50 fans universally said, "Thank god Steve extended his show number beyond 999." They were worried. So they're very relieved. Thank you, on behalf of all of Wisconsin.

Steve: I'm glad I did, and I'm glad Michael got to see his Packers.

Leo: Oh, my. It was a terrible game, but that's okay. We had fun. And if you're an NFL fan, going to Lambeau Field is really kind of going to Mecca. So it was great for all of us. We had a great time, yeah.

Steve: So Leo, if you've been short of sleep lately, once we get into the main topic today, that might be a good time.

Leo: Oh, dear, no. Why do you say that, Steve?

Steve: We're going to talk about, as I was writing this up, I was thinking, oh, wow, I mean, this is really interesting to a subset of our listeners. I mean, the idea...

Leo: Steve, Steve, you're not supposed to say that. You're supposed to say, oh, you're going to want to stay tuned to find out about Encrypting Client Hello.

Steve: It's crucial that we talk about it. But one of the things that I've liked best about the Internet is that it was designed in the beginning by some really smart people who basically kept it simple. I mean, even when we tackled, okay, how does it work, like how does packet routing work, once we explained that a packet has an address, and when it goes to a router, the router looks at it and goes, oh, well, that means I need to send it down this wire, or I need to send it down that wire. And then it goes to another router. And that router does the same thing.

Leo: Yeah.

Steve: I'm going to send it down that wire, or I'm going to send it down that wire.

Leo: Yeah, it's very simple, yeah.

Steve: Yes. It was simple. And even as we've added more things, it's basically stayed simple. Now that changes.

Leo: Uh-oh.

Steve: It turns out that there has always been a privacy leak. Even with HTTPS and TLS and all of our authentication and all of our encryption, there's still been a problem. And here's where it gets - the story turns sad.

Leo: Oh.

Steve: Because to fix it, I mean, to really fix it, we lost simplicity. It stopped being clean in a way that it has always been. So we're going to talk about this today, and we have to because it's going to be the way the world works in the future. But oh, it's not simple. We

lost that elegance. But first - but we're going to get to that. We're going to answer a bunch of questions first because our listeners have said we like the question format. So just how irresponsible have the developers of the most popular email server on Earth been shown to be?

Leo: Oh, boy.

Steve: Bad. What nefarious intent has infiltrated AI dialogue? Windows 11 now supports Passkeys, but what does that mean for the browsers and the add-ons that already do? The tech press is warning about a new password stealing attack against users of public WiFi. 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?

Leo: What?

Steve: Oh, yeah. 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 own websites' content? And I'm not sure why you would want to, but okay. 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 #942 for October 3rd, 2023. Buckle up.

Leo: And prepare to be bored, apparently, a little later on. But, you know what, I know you're not going to...

Steve: No, no, no, no. For the right people, this is one of those where they're going to be glad the propeller on their beanie is as large as it is because it needs to grab a lot of air.

Leo: So this is a challenge. That's good.

Steve: This is a good one.

Leo: Here's a chance for you to stretch your brain a little bit.

Steve: And of course we do have a good picture.

Leo: We do. I am ready for a picture.

Steve: So anyway, this is clever. Some family has three cats. And they thought, hey, that's nice, we'll get this cat food holder that's got three bowls.

Leo: We have this exact product.

Steve: No kidding.

Leo: Note the ears, yes.

Steve: Even with the kitty ears showing.

Leo: Even with the ears. And those bowls. I know this product.

Steve: So anyway, this is a two-frame picture. In the first frame we see the three bowls, sort of in a nice stand.

Leo: Each cat gets their own, yeah.

Steve: Each cat has its own, exactly, his own food. And the caption here is "Developer: Makes a simple, intuitive UI." And then in the second frame we see what the cats have done with this. The white cat is standing behind the bowl on the left, but is eating from the bowl on the far right.

Leo: This is so real.

Steve: Exactly. The black cat is not dissuaded by this. It stuck its head underneath the white cat's tail, and it's decided to eat from the bowl on the left, even though it's standing in front of the bowl on the right. And the tabby is in the middle with its head under both of the cats, the black and the white cats, which are crossed over to the opposing side bowls, and it's eating from the middle. So essentially it's the only one that understood the user interface. The other two not so much. And the result is just sort of not what the designers intended.

Leo: Typical. They're cats. C'mon, man, they're cats.

Steve: And then of course the second frame, where the first frame was labeled "Developer: Makes a simple, intuitive UI," the second frame is labeled "Users." Yeah, exactly.

Okay. So here's the question. What can make things worse for a very widely deployed, public-facing Internet server, like server family, 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? Well, how about that server's publisher ignoring the ZDI, you know, the Zero-Day Initiative's attempts at responsible disclosure of these problems to them for over a year.

Leo: Oh, boy. Yikes.

Steve: And the server in question is the most popular server on the Internet. It's the open source Exim (E-X-I-M).

Leo: Oh, boy. A lot of people use this.

Steve: Yes. Three and a half million people at last count by Shodan.

Leo: To reassure people, our sponsor Fastmail uses Cyrus, not Exim. But a lot of others do.

Steve: Yup, lot of others. So 3.5 million currently exposed online based on a recent Shodan search, most of them in the U.S., I think it's 1.9 million some were in the United States, followed by Russia and Germany in second and third places. Back in June of 2022, ZDI reached out to Exim to inform them of multiple known highly critical problems that had been found by an anonymous researcher. And we're talking about highly critical as in a CVS score of 9.8, which as we know, that's difficult to get. On June 14th of last year, 2022, after asking for and receiving a contact, you know, contact information for the right person to speak to with Exim, ZDI reported the trouble. So that was June 14th, 2022.

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 please resend the reports. On May 10th, ZDI re-sent the vulnerability reports. Then another four months went by until finally last Monday the 25th of, what, September, ZDI again asked for an update while also informing Exim that, you know, we've been patient enough. We're going to publish the entire case as a zero-day advisory in two days, which was last Wednesday. And ZDI has written in their disclosure, they said: "Given the nature of the vulnerability, the only salient mitigation strategy is to restrict interaction with the application." Which is a very political way of saying "Unplug the server now."

Two days later, last Friday, BleepingComputer's headline read: "Millions of Exim mail servers exposed to zero-day remote code execution attacks." BleepingComputer wrote: "A critical zero-day vulnerability in all versions" - all versions - "of Exim mail transfer agent (MTA) software can let unauthenticated attackers gain remote code execution on Internet-exposed servers." Yes, that's all 3.5 million of them.

They said: "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, meaning two days before last Friday when BleepingComputer published this, 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.'"

And BleepingComputer continues: "While ZDI reported the vulnerability to the Exim team in June of 2022 and re-sent info on the flaw at the vendor's request in May of 2023, the developers failed to provide an update on their patch progress. As a result, ZDI published an advisory on September 27th, with details on the zero-day and a full timeline of all exchanges with the Exim team."

Okay. So following ZDI's actions there was some back and forth on the Open Source Security 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.'"

So for all of our listeners, not much imagination is required to know what's going to happen next. The Exim email message transfer agent system is open source and is thus wide open for inspection by anyone. And the ZDI write-up says that the flaw is in a component that handles authentication. So now the world knows that somewhere around 3.5 million of those servers, all of 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, and be glad that you're not running a not-yet-patched Exim server on your network. And watch what happens next.

I have a feeling we'll be talking about this for at least the next couple weeks because, as we know, even if patching was made available, and the people running the servers were notified, there would still be, you know, an exponential curve of patching with lots of machines never getting the message, being on their networks and being left unpatched. The bad guys don't even have to reverse engineer from the binary. They could just go, ooh, good, let's just read the code and find the problem. So that's the world we live in today.

Speaking of BleepingComputer, they carried another piece of news recently, this one with the headline "Bing Chat responses infiltrated by ads pushing malware." And of course Bing Chat is AI-powered now. BleepingComputer explains. They said: "Malicious advertisements are now being injected into Microsoft's AI-powered Bing Chat responses, promoting fake download sites that distribute malware. Bing Chat," they wrote, "powered by OpenAI's GPT4 engine, was introduced by Microsoft in February of 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 their malware. Furthermore, conversing with AI-powered chat tools can instill unwarranted trust, potentially convincing users to click on ads, which is not the case when skimming through impersonal search results. This conversational 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."

Okay. So as we know, 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, no matter what else everyone does. 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 are able to slip past Microsoft's screeners, as Malwarebytes has found is happening, then that's a formula for an updated form of exploitation.

We do have, however, some good Microsoft news. They're rolling out support for Passkeys in Windows 11. Last Tuesday, exactly a week ago on the 26th, Microsoft announced sweeping improvements to the Windows 11 desktop experience. 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. Under Windows 11, Passkeys are created through Windows Hello. Users can manage their saved Passkeys by heading to Start, then Settings, then Accounts, then Passkeys. So there is now a Passkeys entry under Accounts, which is under Windows Settings.

Now, Microsoft says Passkeys on the Windows 11 desktop will work with popular browsers, including its own Edge, and but also Chrome and Firefox. What's unclear at this point is what that means when one of those browsers contains its own support for Passkeys. You know, Chrome, Edge, Safari, and Opera all currently support Passkeys natively. So how does that interact with the underlying OS support for Passkeys? And of course cross-browser support through add-ons is coming soon. So, okay, now there's a third player who's also going to be supporting Passkeys. It's going to be interesting to see how all of this sorts out. And of course, you know, we have the problem that the Passkeys themselves, as we know, are not readily transportable cross-environment. It is possible to create another Passkey in a different environment, but that just creates basically a fork of your logon credentials with separate Passkeys in separate environments. So again, we're going to have to see how this all works.

Two other things, though. 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 in this update to Windows 11. One is enhancements to the built-in Windows Firewall. The other is a new Custom App Control option to ensure that only approved and trusted apps are allowed onto devices to protect endpoints from rogue code. Microsoft said, as part of this, they said: "By preventing 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," they said, "of defending against malware." And I definitely agree. It is annoying in any system that's inherently dynamic to have this since any changes require multiple steps. But dynamic systems are also the ones that are in the most danger because things are changing on them.

So, you know, it's a classic case of you can either have security and jump through hoops in order to allow something that is not currently whitelisted to get trusted and whitelisted. And of course, you know, whitelisting's no good unless you're very careful about that process. Or not. You know, you can just leave your system wide open. So I do commend Microsoft for creating the option for users of Windows 11. It is under any circumstances definitely a good thing.

Okay. Now, here's a case where the tech press got a little overheated. But it's certainly interesting, if nothing else. A team of seven Chinese researchers at three different universities have done some interesting work. And when I began writing this up I used the term "amazing work" rather than "interesting." 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. And as we'll see, it renders it mostly of academic interest. Stated another way: "It kinda worked in the lab."

Despite that, predictably, most of the headline-driven tech press went nuts over this because the research paper, and I guess a lot of the press just read the headline, it was published a few weeks ago titled: "Password-Stealing without Hacking: WiFi Enabled Practical Keystroke Eavesdropping." None of that is true, but makes a great headline.

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, if you change the word "practical" to "barely theoretically possible on a good day when the wind is blowing in the right direction," you know, unfortunately that takes a lot of the punch out of the headline.

Leo: It does, yeah.

Steve: And they do deserve to have some punch because what this group of researchers managed to pull off given a bizarre side-channel is impressive, even if it isn't even remotely practical. And of course we can never fully discount Bruce Schneier's observation that "Attacks never get worse, they only ever get better."

Leo: Other way around. They never get better, they only get worse.

Steve: No, no, no. No, no. Attacks...

Leo: Oh, I hear what you're saying. It depends whose point of view. From the bad guys' point of view, they only get...

Steve: Correct.

Leo: Right, okay.

Steve: Yes, because they're the attackers.

Leo: They're getting better, yeah. If you're the victim, they only get worse.

Steve: That's right, yes. Okay. So the underlying enabling technology that these engineers went for is the result of very 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 mindboggling to imagine that consumers are able to purchase something that does this without even knowing about it.

Modern WiFi access points contain multiple antennas. Individually, each antenna is omni-directional. It sends and receives uniformly in all directions. But collectively, some magic can happen. If the access point wishes, for example, to send a stronger signal to a receiver that's directly in front of it, sending the WiFi carriers "in phase" from its antennas will result in each antenna's carrier, you know, radio frequency carrier wave, summing with the others to produce the strongest signal where they are all in phase, which is either directly in front of or directly 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. That is, off-axis. The carrier waves 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 signals, and it will create "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.

Okay. 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're able to 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 with all this beamforming in real-time. There's a whole other dialogue going on in the background. This metadata is known as BFI, Beamforming Feedback Information. In real-time, WiFi 5, 802.11ac devices, including smartphones, are sending back detailed information about the signal they're receiving from the access point base station to which they're connected.

Now, okay. Remember 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. The term that was coined for this was "Antennagate," which led to Apple's official statement at the time, which read, this is Apple: "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," they said.

In other words, 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, and it's less of a problem for them.

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 to anyone else nearby who might be interested in receiving and interpreting it.

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

If they train, if they highly train a powerful recognition system on a single specific setting and individual, who's not changing the grip on their phone, where the system has already learned to associate the Beamforming Feedback Information-related signals to what's being entered, then under 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 in real time, 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, which were just recently printed: "Exploit steals passwords by tapping into keystrokes." And "New Cybersecurity Threat 'Wiki-Eve' Allows Hackers to Steal Passwords." And finally, "Using Free WiFi? Better Watch Your Passwords." In other words, you don't have to be worried about anything.

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 WiFi settings.

Leo: What a relief.

Steve: You know? Yes. Bruce is right that attacks only ever get better, that is, the strength of attacks only ever increases. But the limitations inherent in this one, to a single previously trained instance, means that it is only of theoretical interest at best.

And I did want to just note that while I was looking up Apple's iPhone 4 statement, I was greeted by Engadget's pop-up, which said: "Review your global privacy control preferences." And it said: "You're using global privacy control." Right. How dare you? They said: "This leads to a lower quality experience on Yahoo! by blocking certain editorial content, including embedded tweets and YouTube videos and" - this is what they really care about - "third-party ads that are relevant to your interests."

Anyway, we've discussed this before where it then goes talking about the technical identifiers which they use. And we looked at those, too, and they're truly horrifying. So anyway, I was offered the option of allow or don't allow. And, you know, I had already got what I wanted, so I said thanks anyway and went elsewhere.

Leo: So why is there no Nobel Prize for mathematics, Steve?

Steve: Okay. So a little bit of context from last week for you, Leo. We had a listener who wrote to me and said that I was incorrect in an earlier statement I had made that no mathematical algorithm could be used to generate truly random numbers. He said he had invented such a mathematical algorithm, I think it was 45 years before, and was surprised that no one had done that since. So I sort of, you know, I mean, I explained why I still believed that it was not possible for a deterministic mathematical algorithm to produce truly random numbers, and suggested that if he actually had such a thing, he could probably win a Nobel Prize for Math.

Anyway, so Jaque Jarnell said: "Hey Steve, quick correction to last pod Episode 941. There is no Nobel Prize for Math." So seeing that, that surprised me since it would seem that there's a lot to math that might be prizeworthy. And there turned out to be a few interesting bits surrounding this. What I immediately encountered was the statement that, on the Internet of course, where everything is true: "No Nobel Prize is awarded for mathematics because a mathematician was carrying on an affair with Alfred Nobel's wife."

Leo: What? Oh, my god.

Steve: Now, that, you know, that would be quite petty, I think, you know, to punish all future mathematicians throughout time. So Snopes, however, disabuses us of that

fanciful notion. They explain: "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 10th of December, 1901, the fifth anniversary of Nobel's death, for achievements in the fields specified by Nobel: physics, chemistry, medicine, literature, and peace. And then a sixth prize was added, a sixth category, the category of economics was added by the Bank of Sweden starting in 1969."

And Snopes continues: "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, one little problem with that. The wife theory is easily discounted since Nobel was never married."

Leo: Oh, well, there you go.

Steve: Yeah. I'll also note that the Nobel Prize Internet Archive has an extensive page addressing this question, 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.

So Wisser tweeted from @wisser, he said: "Hi, Steve. Thanks for the show. Just a short note. The Nobel Prize is not awarded for mathematics. Mathematicians" - as you said, Leo - "can only hope for the Fields medal. Greetings from Stockholm."

Vampire tweeted: "@SGgrc Why not establish a presence in the fediverse? Its open nature seems right up your alley." So just to address that, it's just a matter of being spread too thin and of needing to maintain a presence in too many places at once. 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 to me in helping to move projects to completion. And GRC now maintains web forums. And I still have Twitter, which remains effective. So adding yet another venue to the mix would take away something from the others. If anything, what I would prefer to do is to consolidate, rather than further spread things out.

Someone whose handle is Trunolimit, he says: "Hi. In regards to LastPass, is the email attached to the encrypted blob the only way bad guys know what blob to spend money decrypting?" He said: "When I signed up for LastPass, I made a brand new email that was never used anywhere else." He says: "I don't even get spam to that email." Right, because, you know, you never exposed it apparently.

Okay. The answer is, unfortunately, no. Email is not the only way. For reasons that were never made clear, since it would not 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, could be used to profile users' interest in cryptocurrency-related sites to identify potential higher value targets, where decrypting their vault may reveal something that could be used to drain their money. So given that, it's hard to imagine that the bad guys would not do so after they finish decrypting the vaults of other lower hanging fruit that seem may be hiding some money.

Simon Zerafa tweeted: "Blocking AI scraping and similar from your web sites. Robots.txt and other measures to try

to keep content from AI training models." So Simon tweeted a link to a lengthy post by someone named Neil Clarke which was titled "Block the Bots that Feed AI Models by Scraping Your Website." So I've included the link 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, as we talked about before with Microsoft's Bing Bot. So I'd like to have GRC well represented in the learning models of AI things. But the short version is that the well-known robots.txt file in the website root directories can also be used to block AI bot scraping just as it does for other sorts of web spiders.

You know, 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 their models. Anyway, if you're interested in excluding AI training on your site's content, though it's likely too late for what's already been crawled, there are presumably ways to do that moving forward. I've got a link in the show notes that lists all of the various user-agent strings that can be used if you want more comprehensive anti-AI bot scraping for yourself.

Someone whose Twitter handle is Apples Oranges said: "Quoting Google Bard AI," so he says, quote, this is...

Leo: Okay. Consider the source. It's an AI. Okay, go ahead.

Steve: Uh-huh. So this AI said: "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're often used to solve problems that would be difficult or impossible to solve with deterministic algorithms." Okay. Now, that statement begins with "I personally believe."

Leo: Problem number one.

Steve: Which puts me off, yes, it 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. You know, we are stepping into a weird world here.

Leo: Yeah, we are, yeah.

Steve: Leo, I have an exceedingly, and I mean exceedingly, bright friend of many decades who has been spending a great deal of time chatting with one of the AIs, I don't remember which. And he assures me - and the problem is I really respect this person's opinion. He assures me that over time, with proper grooming, a true personality emerges. And he did tell me that, apparently, praising it helps a lot. Wow.

Leo: My point would be where does that personality come from? It comes from him, the person; right? The person is a - and this is my big problem with a lot of conversation about AI is we humans are applying this layer of anthropomorphism on top of it.

Steve: Yes, that's exactly the right word, "anthropomorphism," right. I mean, you know, cats look like they have something in mind when they're doing stuff.

Leo: Sure. They're thinking, oh, yeah, sure, right. No, they're not.

Steve: It's like staring at you, thinking, you know, basically it's where's my dinner.

Leo: This is how humans work. We do that. But it's not what - it's a machine. Now, what about what it says about non-deterministic algorithms?

Steve: Well, yes. As for whether the phrase non-deterministic algorithm is an oxymoron, I will readily concede that the phrase is used by people who don't consider it to be oxymoronic. But mostly I was just having fun when I was talking about this before, and I believe it's still clear because I didn't say, you know, it was this other guy that talked about a non-deterministic algorithm. I was talking about deterministic algorithms, and I believe it's still clear that any fully deterministic algorithm cannot produce truly random numbers. Which, as I said, was my original statement. It was our listener who introduced the idea of non-determinism about algorithms. And I was just, you know, saying, well, can you have an algorithm that is not deterministic?

Leo: Well, I'll give you an assertion. An AI cannot produce a truly random number, I would guess, I would suggest; right?

Steve: If it were staring at the wall of...

Leo: Lava lamps.

Steve: ...of lava lamps, and...

Leo: Or a capacitor hooked up or, you know, some other physical...

Steve: Yeah. And so if you say that an algorithm is flipping a coin, then okay. But, you know, again, it really comes down to our definition of algorithm. Is an algorithm, you know, one plus one equals two? Or can you say that the algorithm is ask a random number generator for a number? Well, if that's what you're going to do, then I don't think that's an algorithm.

Leo: Well, we know that quantum computing is not deterministic. That's its chief advantage; right? It's neither on nor off. It could be any variety of states. We know

also that fuzzy logic, we talk all the time about using fuzzy logic, or fuzzing, right, to solve security issues. That's not deterministic; right? It's kind of randomized inputs.

Steve: Yeah, although there it doesn't have...

Leo: It's deterministic, I guess.

Steve: There it could be pseudorandom.

Leo: Right.

Steve: But, okay. So, for example, one of the best ways, a little box to generate random numbers is to force current backwards through a diode.

Leo: Right.

Steve: You reverse-bias a diode, which does not - it wants to prevent any current from flowing. What it turns out is that there is some noise of individual electrons just through heat, and, I mean, true quantum-level stuff, crossing over that boundary junction in a diode. You amplify that, and you clean it up, and that is a source of true entropy. And so that's what all the little boxes that say, you know, contains a true random number generator, that's what they have. You can also use a Geiger tube and stick it out, you know, hold it up in the air. And when a neutron or whatever the heck it is flies by, or a charred particle passes, it's like, oh, look, I heard a tick on my Geiger tube.

Leo: These are essentially chaotic inputs, unpredictable inputs.

Steve: Right, exactly.

Leo: Somebody says in the Discord, and I think this is actually a good definition, non-deterministic algorithm would be different outputs for the same input. Which, by the way, most of the time in programming it's something to be avoided. Right? The problem with pseudorandom number generators is eventually they repeat because they aren't really random. But a diode, backwards diode, or a Geiger counter.

Steve: Yeah.

Leo: It's going to be truly random.

Steve: Yeah.

Leo: I guess dice. A die is random. Except there's little influences from the shape of the die.

Steve: Yeah. If you actually, I mean, if it's not perfectly manufactured. And also, you know, I mean, dies have different numbers of divots on their different faces.

Leo: Right. So it's going to...

Steve: So has anyone ever actually taken the time to compensate for that? I don't know.

Leo: Right. No.

Steve: No.

Leo: And we know that you can load die by shaving them, just changing the weight of one side slightly. So that's sufficient.

Steve: Yes. So Leila...

Leo: Go ahead. I love this subject because...

Steve: It is a fun subject.

Leo: And to me, that is the criterion I use to determine between AI and coding. Algorithms are inherently deterministic, in my opinion. But in AI, because it's creating its own algorithms, is non-deterministic in that sense. So if it's machine learning or a generated adversarial network, generated adversarial network or something like that, it's writing its own rules. Not a human. It may still be, frankly, a deterministic algorithm. But we don't know what's going on inside that black box. So from our point of view it isn't. That to me is AI. But, you know, so I'm curious. It's mostly semantics. You know. We know what an AI is.

Steve: Yup. So Leila Burrell-Davis, she tweeted. She said: "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 that maybe Syncthing is not too hard for me. Have a quick look. I'd be surprised if you don't watch the whole thing, 'Syncthing Made EASY.'" Anyway, because I was in the middle of compiling the show, I was unable to take the time to watch. But I quickly scanned the 30-minute video's 165 comments of universally strong praise. Since Syncthing can be initially somewhat off-putting and confusing...

Leo: Oh, this is TechCraft. Yeah, they do good - he does very good stuff. I would trust him, yeah.

Steve: Cool. So I made it our Shortcut of the Week for any listeners who've heard Leo and I because both of us love and use Syncthing.

Leo: Oh, yeah.

Steve: Grc.sc/842, today's podcast number. It's about, I think it's 28 and some seconds long, 28 minutes long. And apparently, you know, it's great.

Leo: Yeah, it looks pretty good. I agree with you.

Steve: So I recommend it.

Leo: With my cursory scan of the thumbnails, it looks pretty good, yeah.

Steve: So FuerstOpus, he said: "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's vulnerable. Thanks, Scott."

And I think that's a very good point. Since the middlebox 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. So I think that's a very good point, and I just wanted to share it with our listeners.

Someone whose - his picture in Twitter is a Cookie Monster. And of course his handle is "Cookies?!" He said: "Heya, Steve. What happens when the person responsible for securing the family's digital life (photo storage, general storage, accounts, et cetera) 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 set up all the accounts, the multifactor authentication, the recovery codes and so on. The other one loves them, so puts up with all the extra hoops that they're being made to jump through, but may not fully understand all or any of it. The issue is the line between ensuring there are no weaknesses in your setup versus 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 and audio books, so unfortunately I had missed the last eight months as I got up to Book 14 of the Silver Ships once you mentioned it!"

Leo: Oh, wow. That's great. That's hysterical.

Steve: And he said: "Plus Leo and his Screen Savers was the reason I asked my parents for Satellite TV when I was 14 in the early 2000s."

Leo: Now I feel old.

Steve: Signed Matt from Australia.

Leo: Yeah. Way to go, Matt.

Steve: So I included Matt's question, even though it's not a new problem, due to its importance, which I think endures. The question, of course, 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. Perhaps we did at some point in the past take a snapshot of our security precautions, passwords, et cetera. Is that snapshot 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. So anyway, just a reminder about that.

Leo: In my desk drawer, in case anything happens to me, Steve, tell Lisa this, I have a piece of paper that says in the case of my death or dismemberment, and inside is all the passwords. I even taped a YubiKey in there, my backup YubiKey in there. But also Bitwarden and LastPass and most password managers have a way to specify an emergency contact, a recovery contact. So I've done that with Lisa and other family members for my Bitwarden. In fact, when we got our new iPhones I noticed you can do the same thing with your iPhone. You can specify an emergency contact, emergency, not - contact's wrong because it's after your death or incapacitation, but an emergency person who has access to your account. And so I've done that also. And in fact I told Lisa and our son Michael about it, and they - we did that all. And I think that - so I think there's an increasing awareness of this.

Steve: Good.

Leo: And you're right, you've got to keep it up to date. Although, if you do it the way Bitwarden and Apple do it you don't have to worry.

Steve: Yes. Right. If you provided means to get in to your master set of secrets, then the changing secrets are always current whenever anyone gets in.

Leo: Yeah, yeah, good. Yeah, Apple calls it Account Recovery Contact, just so you know, Account Recovery Contact.

Steve: Account Recovery Contact. Interesting. It will be interesting to look at how that works, exactly, because...

Leo: This one maybe requires you to still be alive, come to think about it.

Steve: Or maybe it requires you not to deny their attempt.

Leo: That's how Bitwarden works it. It's a dead man switch. So they send an email, and you set the time span, and you say, yeah, if I don't respond in seven days, give her access.

Steve: Let them in.

Leo: Yup.

Steve: Yup. Good. So Mark W. Clemens said: "I received a series of three" - I'm sorry. I received a series of three interesting tweets from Mark W. Clemens. He said, first: "Hi again. Maybe you or Leo could cover some security solutions with Shop Pay (Shopify)." He said: "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." And he says: "(Using my name is fine. I am your age and hard to embarrass.)"

So then I got an update tweet. "Update," he said. "I received this from http://privacy.com." And he said: "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."

And then the third tweet from him. He said: "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 DMs? I think I will join you in leaving Twitter. This was just too coincidental."

Okay, now, this, of course, is all 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 DMs? 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 in the terms of service say? And just to keep some perspective here, Google does something similar, as we know, with our email, which is private only, you know, to about the same degree. Anyway, so who knows? Anyway, just a heads up about something that might be happening in case anyone else cares. I don't know one way or the other.

CPUGuru said: "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.'"

Leo: Yeah. Yeah, sure, yeah, yeah.

Steve: He said: "Had to log in under Chrome to do the export process. Le Sigh," he said. So I included this here because I am 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 their font sizes to other than 100%, it is finished, and it's 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 has not 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 its 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. Unfortunately, reputation is the best defense we have today. And for what it's worth, once ValiDrive has established itself, much as the DNS Benchmark, InControl, and GRC's many other freeware utilities have, it won't cause anybody any trouble. But initially I can expect that it will, and I'll remind everybody about that again next week because what our testers have been testing, you know, versions of ValiDrive, and in many cases, you know, they've like had to fight with Windows in order to get a copy to run, even though they just freshly downloaded it from GRC, and I just built it from the source code. You know, it's like it's minutes old. But unfortunately, that's part of the problem these days.

Matthew N. Dudek, he said: "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 PCs that were used in a medical office I want to give to another organization and want to make sure nothing can be recovered."

Okay. So I poked around a bit to see whether I could determine what Windows is doing. But Microsoft isn't being very clear about that. And I didn't, like, take the time to set up Windows and do a clean-the-drive option and see what is left. So at the moment, of all the things that have been suggested, I really like the idea that was suggested by one of our listeners, 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 pseudorandom data to be written to the entire drive.

Since VeraCrypt has a portable operating mode, it doesn't even need to be installed. It can be run from a thumb drive. And if you just type a bunch of gibberish into the keyboard for your password, and encrypt an existing drive that way, and then throw that password away, you've just filled your drive with absolutely strong pseudorandom data.

Leo: Although this is the argument for turning on file encryption before you use any media.

Steve: Yes.

Leo: And, now, a Mac, it's on by default, FileVault's on by default. I guess BitLocker is not available in Windows Home? Or is it now?

Steve: You're right, I think it's not available.

Leo: Yeah. On both iOS and Android, as far as I know, the phones are encrypted by default.

Steve: Correct. There is on-the-fly hardware encryption on the drive. So it's just a matter of discarding that key and the entire drive is no longer recoverable.

Leo: Whenever - and especially on SSDs we've talked about. But in general, whenever I set up a new drive, I make sure they're encrypted. That way I don't even think about it; right? Because it's just garbage on that platter.

Steve: Right.

Leo: Even the slack space.

Steve: Right. Neil Baldridge said: "Hi, Steve. Over the years there have been various discussions about battery health, charging, et cetera. And I recall you talking about it being best to keep a lithium-ion battery charged to minimize the charging cycles." He said: "Our company has been moving from ThinkPad laptops to Microsoft Surface laptops; and I have noticed that mine, my 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 Security Now!."

Leo: First thing I would say, by the way, is if you keep it plugged in, you're not stopping charging. It discharges a bit, recharges, discharges a bit, recharges. There is no way to not keep a battery from charging by keeping it plugged in. It's going to discharge over time, and you're going to charge it. So his first premise is false.

Steve: Well, so his information's not out of date. What he's referring to is 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 lithium-ion battery is really, really bad for them. So what's happened is that 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 begin resting the battery at some lower charge level which is much 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. So it actually is smarter, if you just generally use your laptop in an at-home, docked, and plugged-in way, setting it up so that the battery is not always fully charged, but is something more like 80 to 85% charged, is better for the battery health.

Leo: No modern device you're buying will let you overcharge a lithium-ion battery, none. In fact, the last time that happened those were those hoverboards. They burst into flames and were immediately banned. Everything you buy today has circuitry to prevent overcharging. You don't have to worry about that.

Steve: Yup.

Leo: But I think it is often possible to fully drain a battery. There's nothing to stop you from doing that, either.

Steve: Correct. Although actually lithium-ion batteries also have a little circuit on the cell itself.

Leo: It'll give up, yeah, yeah.

Steve: That deliberately disconnects it because it does not want to go all the way to zero.

Leo: Right. I think you could trust, basically, I always just tell people trust the manufacturer and use those default settings. Because they know best the hardware.

Steve: Right.

Leo: Don't try to outwit them.

Steve: Right. Although disabling it in order to go to full charge prior to going on a road trip does make sense.

Leo: Yeah. I do that on my EV, yeah.

Steve: Yeah. And this brings us to the smart recommendation of the week. Kevin van Haaren said: "Hey, Steve. I've been thinking about the issue of securely storing things outside of my cloud-based password manager." We talked about, you know, using your password manager to store the things that were relevant to the web. But, you know, unless there's a reason to store other things there, it would make sense to maybe put them somewhere else. Then you're not having the problem of worrying about what else might have been, might be decrypted in the event of a cloud breach like we're now seeing with LastPass. So anyway, he said: "I wanted a product where encryption was the main goal, rather than a compression product where encryption was an add-on." And I had talked last week about using an encrypted archive as a typical solution.

He said: "For this reason I decided to use a password manager, just not a cloud-based one. And one I won't integrate with my browsers." He said: "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. It's been around forever, and started by forking the code base of the even older KeePassX. They even recently had a code audit." He said: "Files in the database aren't compressed, but certs and ssh keys are tiny, and disk space is reasonably cheap."

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

And one final note: Principal Archivist answered the question about mathematicians. He said: "Old mathematicians don't age," Leo. "They just get irrational."

Leo: Okay.

Steve: Okay.

Leo: What is Hello, first of all? I don't know what you're talking about.

Steve: Hello, actually Client Hello is the name of the first packet that our browser sends to a server.

Leo: Oh, okay. The SYN and the ACK. The SYN.

Steve: Well, that's TCP. The Client Hello is TLS.

Leo: Okay, right.

Steve: So after establishing...

Leo: The handshake at the beginning, yeah.

Steve: Exactly, exactly. So 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 network paradigms. Prior to working at Cloudflare, he developed encryption technology for Apple's Internet Services division. He also co-wrote Symantec's Internet Security Threat Report, which we shared for years, and has degrees in both Computer Science and Pure Math. So maybe a Fields Medal is in his future.

So here's what Nick tweeted, and it's the reason we're going to dig into this today. Nick tweeted last week: "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." He said: "Chrome has been testing ECH for months, and is now enabling it by default in Chrome 117. Firefox is not far behind. Cloudflare just launched support for ECH" - again, Encrypted Client Hello - "for all customers. These changes," he said, "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."

Following Nick's tweet, someone immediately replied to his tweet, saying: "What kind of fallout do you expect to see? I'm guessing a lot of enterprise and consumer parental and 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 (Encrypted Client Hello)."

And of course what all of 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 look up the IP address of the domain the user wanted to connect to and make a query. So the browser would initiate then a TCP connection to the remote IP that it had obtained by querying DNS. Then the client, the browser, 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 was requesting and reply with "Not Modified" rather than resending something that the client already had. This improved the web's efficiency. So that's an example of a so-called "query header" that went along with the query.

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, this 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 lookup for the IP of, for example, www.acme.com might return the same IP as a lookup for www.zebras.com. This meant that two different websites were cohabitating at the same IP. And that's what made the "Host" query header then 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 website 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 matching ServerHello TLS handshake packet.

The problem is, in a modern multi-homed environment, where many different possible websites and servers exist, which one 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? Which website? Which domain? 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. And we have no encryption. Like I said, a classic chicken-and-egg problem.

This dilemma was solved by moving the declaration of which server we wanted, which website or domain 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, which opened the TLS handshake, has included a declaration of which website, which domain the TLS handshake should be made with,

which is to say, which server or domain 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 for a response. And anyone - any ISP, any carrier, any government, any censor or malicious intermediary - is able to monitor what amounts to HTTPS and TLS connection metadata.

Okay. 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. Okay. 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. We're going to into this. But beware that, as I said at the beginning, quite unfortunately, we are 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 be getting even 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." And 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 that key 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 text record. So now the web client is able to use the targeted server's IP address to establish the connection and its DNS-published public key to encrypt the ESNI data for inclusion in its initial ClientHello packet.

Okay, well, there are several problems here. The first is the DNS is an unencrypted and unauthenticated protocol. Which means that 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. Okay. But wait. We do have DNS over TLS, aka DoT, and DNS over HTTPS, known as DoH. So believe it or not, the use of some form of DNS connection authentication and encryption is now required as part of this messy solution. I did warn everyone that we had a kludge coming. 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 now shared among all domains that share a common IP address. So a frontend 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 that single IP. So it decrypts the incoming ESNI data, determines the SNI target for the connection, and returns that server's certificate.

Wow. Okay. The problem here, aside from this being a growing mess, which it is, is that the SNI data is the only part of a connection's metadata exchange that's encrypted. In other words, you know, the SNI didn't used to be encrypted. It was in plaintext. So everyone looking at those ClientHello packets go by knew who the user was asking to connect to. So that was encrypted, creating ESNI. And then also necessitating the mess of having DNS publish the public key which was being used to encrypt the SNI data which would then be decrypted by its recipient across however many domains were sharing that IP. On the other hand, only the SNI data of the entire ClientHello is encrypted. It

turns out 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 mess will have been transient.

But to give everyone a feel for it, here is a snippet of what Cloudflare wrote about ESNI, this proposal, nearly three years ago. Cloudflare wrote: "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."

Okay. In other words, this is just an incredible mess. Getting the privacy that we want is really difficult. So the architects of all this set about coming up with a means for encrypting the entire ClientHello packet, leaving nothing of it in plaintext for any snooper to obtain. And to their credit, they also solved the problem which that snippet there at the end mentioned with stale DNS data messing everything up. So how do we get to the title of today's podcast, which is "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 is no really good solution. But stepping back from the details, which is we're about to step into further, I can appreciate that eventually, once all of this has been established, like sometime in the future, and is in place, and is just the way things are done, a major step forward will have been taken toward improving the privacy and security of the Internet. It's just that things had been, as I said at the top of the show, so much simpler and more straightforward until we really required this level of privacy enforcement.

Okay. In any event, ESNI, with all the time and effort that went into it, will eventually be fully replaced by its successor ECH, Encrypted ClientHello. This would close the gap left open, and it will close the gap left open by ESNI. It will protect all privacy-sensitive handshake parameters. Okay. So how does it operate? Similar to ESNI, ECH initially uses a public key distributed by DNS because there's just no other practical way to distribute the key. And it must be obtained using DoH, DNS over HTTPS, for the sake of preventing eavesdropping on that information. And also security.

So we still have the DNS-is-in-the-loop problem. This key is used during the client's initial server outreach. That is, the public key obtained over DNS used during the client's initial server outreach. But ECH, this successor protocol, has added some clever improvements and fallbacks when the DNS key distribution fails, which make the resulting protocol more robust in the face of DNS cache inconsistencies.

Where an ESNI server aborts the connection if the decryption of the SNI data should fail, 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. Okay. So 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?

So now we enter the mega kludge: The ECH protocol actually uses nested ClientHello messages. Unfortunately, moving it into the future, we're not going to have anything being simple here. We're going to have what they call "ClientHelloOuter," which is sent unencrypted in the clear, and now "ClientHelloInner," which is encrypted and sent as an extension of the ClientHelloOuter. That is, that's the way ClientHelloInner is attached. TLS handshakes have a flexible extension mechanism. And so you're able to just, I mean, and that's what SNI is. It's an extension data field added to the Hello. Well, so now this encrypted inner ClientHello will be an extension added to the unencrypted outer ClientHello. If the decryption of the inner ClientHello succeeds, the server will proceed by using the inner ClientHello data. Otherwise, it will use the unencrypted outer ClientHello.

Okay. But hold on. The outer ClientHello was never encrypted. So we can't really use it for anything; right? Unfortunately, that's true. 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, which is that - I've talked about it before in the ESNI context. That is, it's something that answers all of the incoming connections. It's the common server that initially fields, you know, all attempts to connect at that IP. Using the unencrypted outer ClientHello, this ECH service provider signals to the client that its intended destination could not be reached due to a decryption failure. Remember, because if the encryption had succeeded, then the connection would have been made. In other words, the initial handshake's inner ClientHello could not be successfully decrypted.

And while sending that news back to the client, it also sends back the correct ECH public key, that is, it overrides the DNS which the client first attempted. So if that fails, then this ECH service provider sends back the correct ECH public key, which the client can then use to retry the entire handshake. In so doing, it's able to correct the client's copy of the connection's wrong ECH public key, which it would have received from DNS. Or, you know, if Cloudflare is still rotating these public keys every hour, and the client has an ongoing relationship with some server that Cloudflare is hosting, well, that key is going to get stale. So at some point a connection will fail. In the failure, the updated public key is sent back to the client that uses that in order to encrypt the inner ClientHello packet and is able to resume connections until, once again, that key expires.

So there is a great deal of understandable concern over what breakage is going to occur with these changes. ECH attempts, as I said, to hide the inner ClientHello by defining a new TLS handshake extension to contain it. The hope is that random filtering middleboxes along the way, placed who knows where on the Internet, will just overlook any unknown new TLS extension types. But no one knows for sure yet. The people who initially began experimenting when TLS 1.3 was created were quite surprised that so many connections were failing. It turned out that this was due to traffic filtering and analyzing middleboxes which were crashing when they encountered the original unexpected TLS v1.3 packets.

It was only after the 1.3 designers deliberately redesigned their shiny new protocol to much more closely resemble the existing 1.2 that they were then able to get their packets through, basically creating a lookalike v1.3 protocol that was finally able to succeed on the Internet.

So again, who knows what's going to happen once TLS ClientHello's incorporate a nested and encrypted inner ClientHello. 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. Again, Chrome has been testing ECH for months, and is now enabling it by default in Chrome 117, with Firefox to follow closely behind. And as of last Friday, Cloudflare launched their server-side support for ECH for all of their customers.

So it's happening. I don't mean to come off sounding too pessimistic about all this. I'm glad that this work has been done and that the world is moving forward. Unfortunately, it does mean that, you know, the way that all of this stuff once worked, which was so clear and clean and simple and elegant, is going away.

Leo: For good reason.

Steve: It also wasn't fully secure or private.

Leo: Right.

Steve: And it turns out that, exactly, that creating true privacy for a massive global public network is not an easy thing to do.

Leo: Well, as you said at the beginning, none of this was designed to be secure and private.

Steve: Right, there was no...

Leo: And so it's all tacked on after the fact.

Steve: Yup. Yup. And now this is not a tack. This is a box of nails.

Leo: Let's hammer this in. Steve Gibson always makes - see, that wasn't so hard. Aren't you glad you listened? Your brain is now about one-eighth of an inch bigger, full of all that great ECH information. Thank you, Mr. Steve Gibson.