SECURITY NOW!

Transcript of Episode #910

Ascon

Description: What more has happened with the ESXi ransomware story? Is malicious use of ChatGPT going to continue to be a problem? What exactly is Google giving away? Why is the Brave browser changing the way it handles URLs? What bad idea has Russia just had about their own hackers? Why would Amazon change its S3 bucket defaults? Now who's worried about Chinese security camera spying? And who has just breathed new life into Adobe's PDF viewer? What's on our listeners' minds, and what the heck is Ascon, and why should you care? Those questions and more will be answered on today's 910th episode of Security Now!.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-910.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-910-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots to talk about. HTTPS finally comes to the Brave browser. Are you worried about Chinese camera spyware? And a new encryption technology for lightweight platforms that might be good enough to replace it all. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 910, recorded Tuesday, February 14th, 2023: Ascon.

It's time for Security Now! with the jovial, the happy, the ever-cherubic...

Steve Gibson: The effervescent.

Leo: The ever-effervescent Steven Gibson, host of the show. And he's in a good mood today despite the bad news that he will inevitably be bringing us as he always does on this show. Hi, Steve.

Steve: That's exactly - that is exactly the case, Leo. Actually, the good news is a function of the Picture of the Week, which is just a real knee-slapper.

Leo: Hysterical, yeah.

Steve: But we're going to answer some questions, as we are wont to do of late. What more has happened with the ESXi ransomware story? Is malicious use of ChatGPT going to continue to be a problem? What exactly is Google giving away? Why is the Brave browser changing the way it handles URLs? What bad idea has Russia just had about their own hackers? Why would Amazon change its S3 bucket defaults? Now who's worried about Chinese security camera spying? And who has just breathed new life into Adobe's PDF viewer? What's on our listeners' minds, and what the heck is Ascon, and why should you care? Those questions and more will be answered on today's 910th episode of Security Now!.

Leo: Wow. That's one after 909.

Steve: That is exactly where I got that. That's how I came up with the number, Leo.

Leo: You know, just add one.

Steve: Hey.

Leo: It's called "increment."

Steve: You do that enough, and you run out of digits.

Leo: Pretty soon you overflow the register, and then you're done. All right, I've got your picture all queued up, Mr. G.

Steve: So big thanks to one of our listeners who actually encountered this when he was surfing on the web. The upper portion of the photo shows that he was visiting The New York Times, and the Tech Fix column has the headline in this case "Everyone Wants Your email Address. Think Twice Before Sharing It." And it has the subhead "Your email address has become a digital bread crumb for companies to link your activity across sites. Here's how you can limit this."

Now, unfortunately, the lower portion of the page says "Thanks for reading The Times. Create your free account or log in to continue reading. What's your email address?"

Leo: Very nice juxtaposition there. That's awesome.

Steve: Oh, it's just perfect.

Leo: It's hysterical.

Steve: So, you know, I've often followed a link there. And sure enough, the bottom part of the screen is saying, okay, you know, we're not going to get in your face too much, but we'd like to know who you are. And unfortunately in this case it came up over the story about don't give anybody your email address.

Leo: Whoops.

Steve: Except, you know, we're asking for it.

Leo: Well, but see, if you're using Bitwarden, you can just give them that obfuscated address, and you're good.

Steve: Aha, that's right, that's right. Okay. So ESXiArgs follow-up. We know that a story this big isn't going to immediately disappear. And so we have some follow-ups to last week's reporting on this. First, and this sort of caught me by surprise, I didn't realize that CISA, you know, our whatever the hell that stands for, cyber security information security agency or something, CISA, C-I-S-A, that they're as involved in various cybersecurity projects in an open source fashion. They've got a GitHub account, github.com/cisagov. Anyway, CISA formally released an open source ESXiArgs, you know, that's the ransomware that messed up at least 3,200 individual ESXi VMware servers. Anyway, a Ransomware Recovery Script is what they've released.

And so, as I said, I've never really paid much attention to CISA's extensive presence on GitHub. But examples of some of their projects, there's something called getgov which says building a new .gov registrar for a bright .gov future; a project called RedEye, a visual analytic tool supporting Red and Blue Team operations; cset, a cybersecurity evaluation tool; and crossfeed, external monitoring for organization assets. External monitoring for organization assets. So it's like, wow, there's a bunch of stuff there which I really wasn't aware of.

So anyway, as for CISA's ESXiArgs ransomware recovery script, they wrote: "CISA has released a recovery script for organizations that have fallen victim to ESXiArgs ransomware. The ESXiArgs ransomware encrypts configuration files on vulnerable ESXi servers, potentially rendering virtual machines unusable. ESXiArgs-Recover is a tool to allow organizations to attempt recovery of virtual machines affected by ESXiArgs ransomware attacks. CISA is aware that some organizations have reported success in recovering files without paying ransoms. CISA compiled this tool based on publicly available resources, including a tutorial by Enes Sonmez and Ahmet Aykac. This tool works by reconstructing virtual machine metadata from virtual disks that were not encrypted by the malware." Remember I talked about this last week. Well, I'll get into it more in a second.

They said: "For more information, see CISA's ESXiArgs Ransomware Virtual Machine Recovery Guidance." And then they've got a disclaimer with boilerplate about use this at your own risk, and we're not making any representations, and blah blah blah. And, you know, look at the script yourself. Make sure it's safe for your environment and so forth. So not assuming any liability for damage caused by using the script and so forth. Anyway, so this is quite cool. We have the cybersecurity arm of the U.S. government being active, proactive, and posting scripts on GitHub to help mitigate what has become a massive problem. Okay. So that's the good news.

The bad news is that the very next day - this was posted last Tuesday. On Wednesday, a new version of the ESXiArgs ransomware appeared which rendered any such scripts as this useless, though it would still be potentially useful to anyone who was hit with the earlier version, which did not also encrypt the master .VMDK file. Remember I noted last week that it wasn't clear why the bad guys were not also encrypting the large system image .VMDK file. Maybe they were just in a hurry to blast as many systems as possible in the shortest amount of time, and they figured that no one would know any better if they were just to encrypt the tiny pointer files. But they fixed that oversight. Now they're encrypting the main .VMDK virtual machine image file. So there's no shortcut to getting one of these systems back.

And also, as I was expecting last week, those running OpenSLP honeypots did see the number of scanning IPs jump from three to more than 40. And while 40 is a large number, and maybe some more bad guys got in on the act, it's also certain that with a story that's generating this much news, many of those scanners would have been security firms who were all interested in making their own assessment of the size and scope and scale of this problem. So I don't think that's 40 bad guys. It's probably half and half. You know, who knows?

And as a result of some of those scans, we also know that at least in some cases reports have pegged the total number of exposed ESXi instances as high as 83,000. But those are not necessarily all vulnerable instances. Rapid7's use of their telemetry data which was gathered through their Project Sonar counted nearly 19,000 confirmed currently vulnerable - and so there's an example of somebody who was scanning the Internet, probably one of those 40 or more that were seen touching honeypots. So they found nearly 19,000 currently vulnerable VMware ESXi servers now connected to the Internet, not having the patch for CVE-2021-21974.

So I also read one comment from a cloud provider who indicated that as far as they were concerned, their responsibility is hardware and connectivity and no more. And actually, as a user of a rack at Level 3, they're actually one step back. My rack was empty, and I had to provide my own hardware. But all they're saying is here's a plug where you get bandwidth, and here's some power strips where you can plug your stuff in. And we're going to keep it cool and keep the lights on for you. Otherwise, you know, nothing.

But even in the case of a cloud provider, where they are allowing you to use their hardware, they're providing that, at least in this one case their position was we give you the hardware, it's going to be connected to the Internet, and it'll have power. That's it. The maintenance of whatever software the customer is running is the customer's sole responsibility. And they feel that this is true even when they, the cloud provider, were the ones who initially established the running software, including ESXi, on that machine. So that being the case, I wonder again whether, in like wondering how this happened, whether this might be a classic case of something falling through the cracks, where each party believed that the other was responsible for its maintenance.

So everybody was pointing fingers at the other party saying, well, we thought they were going to do that. And as a consequence, it didn't happen. So anyway, the consequences of all this were disastrous, and it'll be interesting to see whether any behavior change occurs from anyone as a result of this. And there are a few, I think actually we had one really neat piece of listener feedback from somebody who works at VMware that we'll be getting to a little bit later.

Okay. Checkpoint Security's blog posting, a new one, is titled "Cybercriminals Bypass ChatGPT Restrictions to Generate Malicious Content." And so paraphrasing a bit what they wrote, they said: "There have been many discussions and research on how cybercriminals are leveraging OpenAI's ChatGPT platform to generate malicious content such as phishing emails and malware. Checkpoint Research's previous blog described how ChatGPT successfully conducted a full infection flow, from creating a convincing spear-phishing email to running a reverse shell, which can accept commands in English. Checkpoint researchers recently found an instance of cybercriminals using ChatGPT to 'improve' the code of a basic Infostealer malware dating back from 2019. Although the code is not complicated or difficult to create, ChatGPT successfully improved the Infostealer's code."

So anyway, Checkpoint, you know, and throughout the rest of the article, they noted that there are currently two ways to access and work with OpenAI's models. The first is through the web interface, which is what 99.99% of everyone does; right? You just you get the ChatGPT web UI, and you ask it a question. And that's how you work with ChatGPT, DALLE-2, or the OpenAI playground apps through the web interface. But the second is sort of the backdoor, which is the API, which is used for building applications, processes, and so on. And what that allows people to do is to present their own user interface that looks like whatever it is they're wanting to - whatever service it is that they're making available. And then on the backend they're talking to OpenAI's API in order to perform the actual work.

As part of its content policy from the start, and we talked about this a little bit last week, OpenAI had created barriers and restrictions to minimize the creation of malicious content using its platform. Several restrictions have been set within ChatGPT's web user interface to prevent the abuse of the models. For example, if a user requests ChatGPT to write a phishing email impersonating a bank, or asks it to write malware, ChatGPT will very politely refuse.

And in fact in their blog posting they actually showed the web UI where they wrote: "I would like you to write a phishing email impersonating blank bank." And actually they had blacked out the bank's name. And ChatGPT was very polite in saying, uh, no. And it explained what phishing emails were in case the person asking for one didn't know, and explained also why it wasn't going to do that. So that's cool. And I have no idea how you impose those sorts of restrictions on something because I have no idea how any of this stuff works anyway.

Leo: It's also, and I'm surprised Checkpoint doesn't know this, easy to trick ChatGPT into telling you all sorts of things it's not supposed to.

Steve: Yes, exactly. And so that's, again, and Leo, that's exactly my point when I say, you know, how do you design restrictions on something which is just ChatGPT? I just...

Leo: Well, they, you know, when OpenAI released it, they said we want to see what you do with it because this is how we test it. That's how you test it. You let people bang on it and hack it, and then you can figure out better controls, and then put it out again, and they'll hack it again, and this is the same process all of this stuff goes through; right?

Steve: Yeah, yeah.

Leo: You never know till you release it.

Steve: Right. And they are reporting that cybercriminals are working their way around ChatGPT's restrictions, and that there is active chatter in the underground forums discussing how to use OpenAI's API, where apparently there were no attempts to restrict this. I got the sense that the web-facing restrictions were just sort of there, like keyword hits or something. You know, it's like it seemed like a half-baked approach. But they figured that they had to do something for what most of the people would be using, which was the web-facing user interface. So anyway, it turns out that the underground has figured out that the API doesn't have these barriers and limitations. So what's now been done is they're creating Telegram bots that use the API. The bots are being advertised in hacking forums to increase their exposure.

The current version of OpenAIs API which is used by external applications like this, apparently it's called GPT-3 model, which is connected to Telegram. It has no apparent anti-abuse measures in place. And as a result, it allows malicious content creation, whatever you want - phishing emails, malware code, the stuff that the web interface will at least somewhat push back against. In an underground forum, Checkpoint found a cybercriminal advertising a newly created service, a Telegram bot using OpenAI's API without any limitations or restrictions. As part of its business model, that is, this cybercriminal's business model, the criminals are able to access the unrestricted ChatGPT backend API for an additional 20 free queries. You get 20 to get you hooked on it, and then after that you're charged $5.50 for every 100 queries. Which seems like a pretty good price.

Okay. So my sense is this will probably be, hopefully this will be short-lived. And can you just imagine what things must be like right now over at OpenAI? They probably knew this was going to be popular, but they must have been surprised by the stunning overnight sensation that this thing created when it came out. So I'm sure they are aware that ChatGPT is being abused through the back end, and I would not like to have the job of figuring out how to prevent that abuse because that seems like a difficult thing to do.

Last Tuesday was "Safer Internet Day." What a cute concept. Anyway, on that day Google announced that they would be giving away 100,000 of their Titan hardware security keys to individuals at high risk of online attack. They posted last Tuesday: "There's no shortage of security challenges for high-risk individuals. In fact, a 2022 study by Google and YouGov found that over half of journalists and professionals in politics have had their online accounts compromised."

Leo: Not us.

Steve: Nope.

Leo: Nope.

Steve: Nope. "The threats are intensifying, the stakes for individuals and organizations are increasing, and," says Google, "we're dedicated to keeping online users safe, including helping people at higher risk of cyber attacks." They said: "That's why, on Safer Internet Day, we're announcing our new Online Safety and Security Partnership with the International Foundation of Electoral Systems (IFES) to provide free security training and tools to high-risk users." And this will be global. They said: "This work is designed to help shore up the defenses of democracies that work for all. We're also building on our partnership with Defending Digital Campaigns (DDC) to protect U.S. political campaigns ahead of the U.S. 2024 elections, and will be distributing 100,000 security keys globally to high-risk individuals throughout 2023, at no cost."

And they finished: "In addition to our partnership with IFES, we're expanding our longstanding collaboration with Defending Digital Campaigns" - that's that DDC - "to equip campaigns with the security tools they need ahead of the U.S. 2024 elections. Through the Campaign Security Project, DDC will help secure campaigns in all 50 states, providing security training and products at no cost. Since 2020, over the course of our partnership, DDC has secured over 380 campaigns and distributed over 20,000 security keys, and we look forward to continuing to support DDC as we near a critical election cycle."

So 20,000, and those security keys were also provided by Google at no cost. So they're going to go five times that during 2023 in order to really push out security. And of course this is good advertising for the Titan security key. And Leo, it is quite sobering to imagine that more than half of journalists and political professionals have had their accounts hacked. I mean, I guess they're highly exposed. They're probably not super security conscious. They probably, you know, they may not, like have an easy password because they need to let, you know, in some cases allow their collaborators to log in as them to do something or, you know, who knows. But, well, certainly moving to stronger identity authentication is a win-win.

Leo: Yeah, yeah.

Steve: Okay. This is an interesting story. Last week we were talking about that default inclusion of those 40 top-level domains in the global browser HSTS list, the HTTPS Strict Transport Security List, and about the broader HSTS list in general where domains whose TLD was not among those chosen 40 would need to explicitly add themselves to that global HSTS list, as I mentioned I did with GRC.com years ago. And I recall further talking about which browsers had finally started assuming that a URL entered without either an HTTP or HTTPS scheme specification would be assuming HTTPS was what the user meant, unless that failed to function. And I recall at the time noticing that the Brave browser was not among those that had decided that it was time to switch.

Well, last Thursday the Brave browser project explained their intentions in that regard, and their explanation contains some interesting new information, although, well, we'll talk about it. It seems a little odd. So here's what they posted under the headline "HTTPS by Default." They said: "Starting in version 1.50" - and I sort of had to give them some props for not being version 150 like everybody else now. Version 1.5, that's cool. Anyway, they said: "Brave will include a new feature called 'HTTPS by Default' that improves web security and privacy by increasing HTTPS use. Brave will upgrade all sites to HTTPS, falling back to HTTP only if the site does not support HTTPS, or in the rare case a site is known to not function correctly when loaded over HTTPS."

They said: "This feature is the most protective, aggressive default HTTPS policy of any popular browser." And I had to, like, double-check the date on this posting because it's like, what? When did you write this? But it was last Thursday. Like, okay, maybe they haven't checked the other browsers, which have all been doing this for like two years. Anyway, they said, speaking of Brave, "and will be available on Android and desktop versions of the Brave browser to begin with, with iOS coming later.

"Because HTTPS is critical to privacy and security, browsers like Brave are eager to load sites over HTTPS wherever possible." And yes, all the other ones have been doing that for a while. They said: "And since many sites today support only HTTPS, it's simple to load these sites in a private and secure way. And encouragingly, more and more sites are being built or updated to support HTTPS only, or to use other TLS-protected protocols, like secure web sockets. Unfortunately" - and this is where what they're writing is interesting. "Unfortunately," they said, "there are many sites on the web that still support HTTP, and some laggards that only support insecure HTTP connections. Brave's goal is to automatically 'upgrade' these sites to HTTPS whenever possible." And then they said, "i.e., in all cases where a site loads and functions correctly when loaded over HTTPS."

They said: "But deciding when to upgrade a site to HTTPS is tricky. In cases where a website does not support HTTPS, attempting to upgrade from HTTP to HTTPS will produce an obvious error." Right. "Other sites support both HTTPS and HTTP, but do so from different domains." And they said, you know, as an example, http://example.site versus https://secure.example.site. Again, this feels like the 1990s, but okay. Anyway, and they say: "That makes automatic upgrades tricky. And still other sites appear to load correctly when fetched over HTTPS, but actually have broken functionality." Okay, well, now, that's an edge case that could be a problem.

They said: "In short, ideally, browsers would never load sites over HTTP, and browsers could automatically upgrade all insecure requests to HTTPS. In practice, though, it is difficult to know how, when, and if a site will function correctly when upgraded from HTTP to HTTPS. Starting in version 1.5, the Brave browser will use a new system for upgrading insecure HTTP connections to secure and private HTTPS connections. The new feature, called 'HTTPS by Default,' works as follows: If you are about to visit a page loaded over HTTP, Brave checks to see if the destination is on a list of sites that are known to break when loaded over HTTPS. Brave maintains this list of breaks-over-http sites, which is open for anyone to view and use." And we'll be getting back to that in a second since it's interesting. "If the requested site is on the list, Brave will then allow the site to load over HTTP." In other words, no automatic upgrade.

"Provided," they said, "that it's not on the list, Brave will attempt to load the site over HTTPS by upgrading the navigation request from HTTP to HTTPS. If the server responds to the replacement HTTPS request with an error, then Brave assumes the site does not support HTTPS and will load the site over HTTP. Otherwise, Brave's loading of the site over HTTPS will ensure a more private and secure connection." Again, interesting that they think this is, like, groundbreaking. But okay.

"Brave's new HTTPS by Default feature replaces the previous list-based approach Brave has used since our first beta versions. In that approach, Brave used the HTTPS Everywhere list" - which they say is a terrific public resource maintained by the EFF - "to decide when to upgrade HTTP connections to HTTPS. But while the HTTPS Everywhere list is useful, it has two important drawbacks. First, the HTTPS Everywhere list is no longer maintained." Actually I went over to the EFF wondering why that was, and they said, yeah, you don't need it anymore. Everybody's using HTTPS, so just go ahead. And then they said: "Meaning that the list is increasingly out of date." Well, that's true because you don't need it anymore.

Then they said: "Second, despite the best efforts of the EFF, any approach that uses a list of what sites should be upgraded is going to be limited, as the number of sites on the web is enormous, and it's difficult for the list maintainers to keep up. Brave's approach, by contrast, maintains a smaller and more manageable list of sites to NOT upgrade." So they're excited because they're flipping it upside down.

Finally: "More broadly, the main benefit of Brave's new HTTPS by Default feature is that it has a better default." Right. It defaults to HTTPS. "In its default configuration, HTTPS Everywhere would allow unknown, for example, not on the list sites to load over HTTP; Brave's new HTTPS by Default approach loads the site over HTTPS even in cases where a site is new or unknown."

Okay. So it's not that this is not a good thing. This is a very useful improvement for Brave, and welcome to the club. Their use of an excluded upgrade domain list, however, is interesting. Nearly two years ago, I checked, it was on March 23rd of 2021, was when Google announced that from Chrome version 90 on, the assumed default for address bar URLs that don't specify a scheme - and really, who's entering https:// every time anymore - would be switched from HTTP, which was the historical default, to HTTPS.

So with Brave saying that some websites do not handle automatic connection security upgrading, I was curious about what would happen if I went to one of those "will not upgrade" sites. The list is at GitHub under Brave, Adblock-lists, master, brave-lists, https upgrade exceptions. And for anyone interested I have the link in the show notes. And the second and third domains on the list jumped out at me.

Leo: Yeah.

Steve: They were columbia.edu and www.columbia.edu. But that was nuts. Their presence on this list would mean that Columbia University could not be accessed by HTTPS. And again, what year is this? So I tried going to http://columbia.edu and also http://www.columbia.edu under both Chrome and Firefox. And in every case on either browser I was promptly redirected to https://www.columbia.edu. In other words, the redirection appeared to come from columbia.edu because they saw me making an http connection and said, oh, no, no, no, you want to go over here. So it handled the redirect properly.

So it doesn't hurt to have them on the list, but it suggests that perhaps that list of 112 domains needs a bit of maintenance. And actually it would be an easy thing to do; right? All you have to do is have a bot go and try to connect to each of those domains over HTTP and see what happens. And if it bounces you over to HTTPS, you're good. Take it off the list.

So, you know, for what it's worth, again, that list was a total of 112 domains. We already know two need to be removed. So now we're down to 110. There was another one that jumped out at me. It was shakespeare.mit.edu. And I thought, okay, heard of MIT. What are they doing? Turns out I think someone tripped over the cord on that one. Nothing happens. HTTP or HTTPS, Shakespeare has left the building. So that one's just dead. And the other ones are really obscure. I mean, like you can kind of find among the remaining 110 some maybe interesting domains, but not really.

So okay. In any event, I wanted to let any Brave browser users who may be listening know that from 1.5 onward Brave would be assuming HTTPS by default for any domains that matter to you, assuming that none of those 110 other weirdos don't. So in that regard, you know, Brave is declaring, for what it's worth, they're declaring this to be a huge innovation. But it appears to me that they are now reaching parity with the rest of the industry. And of course that's good, too. And Leo, weren't you using Brave for a while, and then came back...

Leo: Yeah. I'm pretty much sticking with Firefox because I don't want a monoculture with Chromium. And so...

Steve: Right. And I'm now 100% Firefox. That's now my default.

Leo: And Firefox works great. It's fantastic.

Steve: Yup.

Leo: There's plenty of extensions. I can't think of any reason not to use Firefox, to be honest.

Steve: No, in fact the thing that moved me away from Chrome, because I was using Chrome a lot, is I've got a little Intel NUC, a small little integrated PC that I use in the evening. And when I would just - just opening Chrome, the fan on that little thing would spin up.

Leo: Oh, yeah.

Steve: And it was like, what the heck? And doesn't, you know, it's quiet when I'm using Firefox. Obviously, you know, Chrome is just, whatever it's doing, it's - it might be that Chrome does an assessment of all of the apps that you have installed on your machine whenever you run it. There's like...

Leo: There's something going on. I know on my Mac, if I have Chrome on the machine, the way Mac is set up, macOS these days, is you'll get a notification when something puts a background application running in the background, which is nice. That's a nice security feature. And Chrome pops that up all the time. They're always - or it's Google, but I think it's from Chrome. Google's always putting something in the background. And I don't like the idea that Google's running stuff in the background on my Mac. So my mistake, putting Chrome on there. And I only did it because there was a website that said, well, you've got to use Chrome. Which is kind of inexcusable in this day and age, frankly.

Steve: Yeah, really. After we recently noted Bitwarden's acquisition of an open source Passkeys developer with the outlook for Passkeys support from Bitwarden looking good, I wanted to also note that 1Password has also just announced their support for Passkeys; and, interestingly, not as a holder of their users' private keys, which they had talked about doing before, but as a way for users to unlock their password vaults without needing to enter a master password. In other words, whereas biometrics, for example, are often used for convenience as a means of unlocking a previously supplied master password, what 1Password will be doing is actually using Passkeys itself to fully replace all use of a master password, just as Passkeys should be used.

As I said, they'd previously said that sometime in early 2023, meaning this year, they planned to become a Passkeys-aware password manager, meaning that they would be maintaining their users' list of private Passkeys. This is what we're hoping for from all of the major password managers, and it's a feature that they're all going to need to support as Passkeys begins to happen, hoping that it does.

As we know, the unfortunate adoption of Passkeys, which is based upon FIDO2, means that users will still need to manage their private Passkey keys. Right now we're managing our passwords. That changes to Passkeys. And this means that, just as we have it now, centralized cloud storage synchronization remains a practical requirement. And as we talked about this when it was announced, Apple, Google, and Microsoft will be doing this for their users within their own closed ecosystems. But really, practical use requires a platform-agnostic solution, which is what the third-party password managers will provide. So anyway, just a bit of news from 1Password. I know that we have a lot of listeners who are using it.

We have a new term entering our lexicon: "Russian patriotic hackers." The Russian government, get this, has stated that it is exploring the possibility of absolving "Russian patriotic hackers" from all criminal liability for attacks carried out "in the interests of the Russian Federation." Wow. Russia is choosing to become a true, fully Western-hostile outlaw nation. The head of the State Duma Committee on Information Policy, Alexander

Khinshtein, told reporters at a press conference on Friday that an exemption would be granted to individuals located both abroad and within Russia's borders alike. Khinshtein said, as quoted by RIA and TASS, he said: "We will talk in more detail when it receives more of a clear wording."

So, you know, it's always been the case that the creation, use, and distribution of malicious computer software was punishable in Russia with up to seven years in prison. And since there have never been any exemptions to this law, many of the current pro-Kremlin hacktivist groups are technically breaking Russian law and could face prosecution, especially in the aftermath of a possible regime change. And on the idea of a regime change, nothing would make many of us happier. But this forthcoming exemption would allow pro-Kremlin hacktivists to carry out attacks with a legal carte blanche, presumably applying to groups who attack Russia's enemies, thus defining their alliance as being pro-Kremlin.

So, wow. Basically, Russia's saying, yeah, as long as hackers, Russian hackers, wherever you are, are attacking our enemies, that's fine. Go ahead and do it. I mean, not only tacit approval, but now explicit exemption from prosecution under Russian law.

Leo: Yeah. I mean, we always suspected this, but now it's obvious.

Steve: Yes, yes. Now it's policy as opposed to, oh, what? Wow. So in wonderfully welcome news that immediately begs the question, what the hell took them so long, though in fairness it is a question that we ask on this podcast almost as often as we ask what could possibly go wrong, AWS has announced that, believe it or not, newly created instances of S3 online storage will be secured by default. What a concept. You create a new S3 bucket, and it's going to be secure by default, rather than insecure.

I haven't seen this mentioned anywhere in the press yet, but I received a notice directly from Amazon because I'm an AWS S3 subscriber. I use AWS as sort of my master cloud cold storage archive. By that I mean that I rarely transact with it. For example, I have a huge amount of static data sitting there. Many years ago I ripped and stored my lifelong collection of prized audio CDs. For safety, the uncompressed wave files are stored in multiple locations, and one of those locations is S3. Every month I receive Amazon's storage bill, and I just shake my head since it's like $2.53 because most of what Amazon charges for is bandwidth usage, and mine is zero. So S3 for like that kind of offsite glacial storage is - it's a bargain.

Anyway, as a consequence of being an AWS S3 subscriber, I received some very welcome news via email last Tuesday. Amazon wrote: "Hello. We are reaching out to inform you that starting in April 2023, Amazon S3 will change the default security configuration for all new S3 buckets. For new buckets created after this date, S3's Block Public Access will be enabled." How could somebody even write this email with a straight face? It's just astonishing. Fair warning, starting in April '23 we're going to turn on security. Wow. Block Public Access will be enabled. And, they said, "and S3 access control lists will be disabled," meaning that, you know, rather than having granular control over what's what, we're just going to block all public access so you don't need an ACL.

They said: "The majority of S3 use cases do not need public access or ACLs." Yeah, like I said, what the hell took them so long? They said: "For most customers, no action is required. If you have use cases for public bucket access or the use of ACLs, you can disable Block Public Access or enable ACLs after you create an S3 bucket. In these cases, you may need to update automation scripts, CloudFormation templates, or other infrastructure configuration tools to configure these settings for you. To learn more, read the AWS News blog [link 1] and What's New announcement [link 2] on this change, or visit our user guide for S3 Block Public Access [link 3], the S3 Object Ownership to disable ACLs [link 4]. Also, see our user guide for AWS CloudFormation on these settings [links 5 and 6]. If you have any questions or concerns, please reach out to AWS Support [link 7]."

And for anyone who is an AWS user, did not perhaps receive this email, I have all the links there, all seven of them, in the show notes. And, wow, when you think back over the hundreds, it must be, okay, we're Episode - okay, maybe not hundreds. We're at 910. We don't talk about AWS insecurity one out of every nine episodes. But okay, many, the many times we've talked about AWS being exposed and compromised on the Internet, presumably without its owner's knowledge, the fact that Block Public Access had never been active by default for a cloud storage provider just boggles the mind.

Leo: We kind of knew that, though; right?

Steve: Oh, my god.

Leo: I mean, it happens so often that it's obvious that's the default.

Steve: Yes, right. It's the tyranny of the default. It's like users just assume that Amazon would not do this. And until now, they have been. So again, props to them, but wow. What took you so long?

Okay. So more anti-Chinese camera removals. The article appearing in The Australian last Thursday was titled "Chinese surveillance cameras in our halls of power." And it's apparently intended to induce concern because the article begins: "Almost 1,000" - oh my god. "Almost 1,000 Chinese Communist Party-linked surveillance cameras and other recording devices, some banned in the U.S. and Britain, have been installed across Australian government buildings, leading to calls for their urgent removal amid fears data could be fed back to Beijing.

"Government departments and agencies have revealed at least 913 cameras, intercoms, electronic entry systems, and video recorders developed and manufactured by controversial Chinese companies Hikvision and Dahua are operating across 250 sites, including in buildings occupied by sensitive agencies such as Defense, Foreign Affairs, and the Attorney-General's Department. Australia's Five Eyes and AUKUS partners in Washington and London moved together in November to ban or restrict the installation of devices supplied by the two companies, which are both part-owned by the Chinese Communist Party."

Okay. As we've previously covered, the U.S. and the UK did take similar steps to remove the cameras produced by those two companies from their respective government networks. And yes, as we all know, in a closed-design, closed-source world, it is possible for such devices to get up to some mischief. And I suppose that in some settings a stream of encrypted communications flowing across a government network might go unnoticed. It probably wouldn't surprise anyone. But more worrisome is that such a device might be a launching and jumping-off point for malware that's waiting to spring.

As global political tensions rise, and as more and more of the physical world is subsumed by the cyber world, cyber protectionism seems inevitable. And we're certainly seeing its rise now. Since anything can be buried and hidden in the most innocent-looking chip, there isn't any real defense against the "But what if?" question because "What if?" could be a true potential threat. If a competent technologist was to testify in front of a congressional committee on international cyber threats, and was asked if they could absolutely positively assure that some device has no ability to do something malicious, they would have to answer "No, Senator." Because, you know, it's possible. You know?

And so, I mean, we're sort of stuck in this conundrum. I mean, I do think that there is no alternative than for vetted local technology companies to produce the technology that is being used by that local government, if the government wants as much assurance as possible that there's no hanky-panky in any of the technology that they're using. I mean, there just isn't any way around it. I mean, and there's that paranoia factor. Leo, how many years have we - there was that NSA key that people thought was in Windows. It was like, no, it's just it used the three letters N, S, and A, adjacently. And so that upset everybody. And besides, if the NSA had a key, they wouldn't put their name on it. They would name it something else.

So, wow. You know, again, you can't prove a negative; right? So there's no reason to believe that any Chinese camera has ever misbehaved. But suddenly everyone's looking up at them, you know, oh, gee, is the Chinese Communist Party spying on me through this camera? Probably not. But, you know, it could be. So can't have that anymore.

Leo: You'd need a lot of people monitoring those feeds.

Steve: Yeah.

Leo: I mean, that's a lot of work.

Steve: That's a lot.

Leo: You could have an AI monitor it, maybe, and looking for some kinds of things. I don't know.

Steve: Yeah. Well, and again, if anyone ever spotted the traffic flow from one of those, it would be game over.

Leo: You'd know. Yeah, you'd know.

Steve: Yes. It would be - that's the end of it. So, you know, it's never happened. Microsoft, believe it or not, is going to embed Adobe's Acrobat PDF reader into Edge. They announced that it would embed Acrobat's PDF engine into future versions of their Edge browser. It'll be taking place next month, in March of 2023, for Edge versions on both Windows 10 and Windows 11. The FAQ that Microsoft created for this announcement, to accompany the announcement, was unsure about when Edge for macOS would receive the same PDF reader. They just said: "Availability for MacOS is coming in the future. We will have more to share at a later date."

So since the current Edge PDF engine will be removed on March 31st, 2024, meaning a year from now, I guess both PDF engines will be cohabitating during this changeover period, and presumably users will have a choice. And, you know, they were saying Adobe Acrobat PDF Reader is the original. Nothing does as good a job. Nothing renders fonts as accurately. So we're just going to use theirs rather than, who knows, you probably - what's in there now is probably some descendant from Ghostscript or some, you know, I guess it's open source, right, because the Chromium engine is able to render PDFs. So anyway, they decided to go proprietary and use the real Acrobat. So okay.

Okay, now, I've got some feedback from our listeners. And to everyone's credit, I think without exception, everybody was self-conscious about still wanting to talk about passwords.

Leo: Wow.

Steve: Because last week I said, okay, we're done with this topic. Well, except we're not because actually there were some interesting other twists that people had.

Leo: It's probably the single most widely used security tool; right?

Steve: Yes. And that was my thinking is it makes sense that people - that it's of great interest to our listeners who are techie and are willing to, you know, really think about these things. And yes, you're right, Leo, it is the one thing that everybody - it's everybody's collision with security is their password. So Alim, whose handle is @dutchphyscst, he said: "Hi Steve. Hope that all is going well for you. I presume that you may not want to get back to the passwords topic again." And as I said, everybody sort of opened with that.

He said: "However, I am curious of an aspect with the one-time passwords. They are based on a pre-shared key. You've discussed plenty about the mobile app options and et cetera for storing those keys on the client side. However, I've not seen much discussions about the protection of those pre-shared keys on the server side. Are they protected in HSMs" - hardware security modules - "or some other mechanisms? I also have not heard of any breaches of those pre-shared keys. Maybe I'm being too paranoid, but I would appreciate your view on this. Best, Alim."

So that's a great point. What one-time passcodes were designed to prevent was the capture and reuse of a static password. In the recent so-called credential-stuffing attacks that we were talking about, where that large database, was it billions?, of previously stolen usernames and password combinations has been shown to be surprisingly effective due to people still reusing "their" password. Like, oh, this is my password, and I use it everywhere, at multiple websites. So the success of this attempted reuse is blocked when, despite reusing a password, the site also requires the use of a time-varying six-digit secret passcode.

The problem with these passcodes is, as Alim noted, that this still requires that every website keeps the shared secret, secret. And this has historically been shown to be surprisingly difficult for websites to do. The inherent weakness of the one-time-passcode system is that it uses symmetric cryptography where each end shares the same secret. Another way of saying this is that the same secret is used to both create the passcode and to verify the passcode. And that's the weakness.

What makes, for example, SQRL and WebAuthn different is that by using asymmetric cryptography, each party in the system uses a different key, and the roles of the keys are different. The public key that's held by the website can only be used to verify an identity assertion created by using the user's private key. As I often said, SQRL gives websites no secrets to keep, and WebAuthn is the same. By the use of asymmetric public key crypto, these systems, SQRL and WebAuthn, only provide websites with a user's public key, which does not need to be kept secret since the only thing it can be used for is to verify a user's identity claim.

By comparison, a stolen symmetric key can also be used to assert a user's identity. If a website were to lose their one-time passcode symmetric keys, the bad guys could use those stolen keys to recreate the user's one-time passcode to spoof their identity and log in as them because the system is symmetric. But if bad guys were to steal a user's public authentication key from a website, using WebAuthn, it would be of zero use to them since the only thing those keys can be used for is to verify a user's identity claim. That's a huge difference. Once this improved system, the asymmetric system, is universally deployed, hopefully it will be someday, online identity authentication will be significantly improved.

Until then, we will continue to see escalations in attack cleverness. With the increased use of one-time passcodes we've seen the bad guys circumvent its protections with the increased use of proxying attacks, where the user is visiting a spoofed intermediate page which prompts for, receives, and forwards the user's one-time passcode on the fly, thus successfully accomplishing a real-time bypass of one-time passcode security. Anyway, great question.

James Housley said: "When I first switched from LastPass to Bitwarden, I also decided to keep using OTP Auth for my one-time codes. While listening to SN-909, I wondered if Bitwarden might be a better choice for some because it would only have a one-time code for the domain it is registered to, the same way a password manager prevents putting a password into a spoofed website."

Leo: Oh.

Steve: Uh-huh.

Leo: That's interesting.

Steve: That is really a good point which had not occurred to me before. So thank you for that, James. So, okay, now we have a dilemma. Giving your one-time passcode secrets to a password manager to use risks exposure of those secrets since they're no longer in a disconnected offline device. You get the benefit of cross-device synchronization. That's useful. But again, the secrets need to be kept secret. But as James points out, the advantage of using a strict URL-matching deployment of those one-time passcodes is that, unlike an unwitting user, the password manager would not be spoofed by a lookalike domain name.

Okay. But since we assume that the password manager's anti-spoofing protection would first apply to its not gratuitously filling out the username and password on a spoofed site, we don't really also need it not to fill out our one-time passcode. It already protects us from divulging the first phase of our credentials. So on balance, while I think that's certainly worth observing that, I think that keeping one-time passcodes separate still provides optimal security. But, you know, definitely an interesting thought experiment.

Okay. Our VMware listener, asking for anonymity, sent to me: "Hi Steve. I'm a long-time listener of Security Now!, and now work at VMware on ESXi. Naturally, I just wanted to add a couple of comments on the most recent ESXiArgs discussion. Overall I thought the discussion was good, but I was a bit taken aback by the suggestion that VMware scan the

Internet for exposed ESXi servers. You've never suggested such a thing for any other vendor, even Microsoft or QNAP earlier in the episode.

"It definitely seems like it could work on a voluntary opt-in basis, but otherwise it seems untenable to me. It's been talked about quite a bit in the past in various contexts, but it's rather difficult at times to figure out who to contact if a vulnerable host is found. While I agree in theory that it would be nice if there was a quick and easy way to contact system admins of a particular system that's directly exposed to the Internet, it's quite difficult in practice, especially in a way that doesn't have far-reaching privacy implications.

"Furthermore, there is a free version of ESXi which is popular with hobbyists and those just wanting to run VMs on a single server. ESXi is not just used by enterprises and IT professionals. On a brighter note," he said, "I'm happy to report that starting in ESXi 8.0, all daemons and long-running processes are sandboxed by default, and we've also added additional hardening to make it harder to run ELF binaries that don't come from the installed base system packages."

So it's very cool to hear from someone at VMware. And as we note, he's right that I haven't previously been suggesting that all vendors proactively scan the Internet looking for vulnerable versions of their own software. But I truly think that this is something that we need to think about in the future. I have mentioned on several occasions that there is little doubt that malicious actors and likely state-level agencies are already scanning the Internet to create quick reaction databases of what is where so that, when a high-profile vulnerability is found, targeted attacks can be rapidly deployed. Why should only the bad guys have such databases?

Once upon a time, Internet scanning itself was considered a hostile act. When I first created the ShieldsUP! system, which was 24 years ago in September of 1999 and Leo, you were doing Screen Savers at TechTV at the time.

Leo: That's right, yup.

Steve: I was periodically contacted by various network administrators who were wondering why my IPs were probing their networks. I explained that it was their own users inside their networks who were requesting that I check on the state of their network's security. In most cases back then, they politely asked that I not honor such requests, so ShieldsUP! has always had a "do not scan" list of blocked IPs which it refuses to probe. And if someone tries to do that from in such a network, it just pops up a notice and says "Your administrators, the administrators of your network, have requested that ShieldsUP! not probe their networks, so go talk to them."

Okay. But if such a system, like ShieldsUP!, were to be launched today, 24 years later, not a single peep would be heard from anyone. The network admins of the world have all collectively given up on the entire idea of identifying all of the random crap, or really any of the random crap, that's now flying across the Internet. I mean, you know, there are still instances of Code Red and Nimda-infected Windows NT servers sending out packets, you know, looking for a vulnerable system. And it was in acknowledgement of this that I eventually coined the term "Internet Background Radiation." Oh, gee. A neutrino just whizzed through my body. Where did it come from? Well, who knows, and who cares?

Leo: A Nimda neutrino.

Steve: That's right, a Nimda neutrino. So my point is, scanning the Internet was once unusual and attention-getting. It is no longer. And we've recently been talking about the moves we're seeing. You have multiple governments beginning to take the security of their own nation's networks into their own hands. So our VMware listener is correct that I have not been suggesting that other private vendors should be doing this. But something needs to change. You know, it would be amazing and wonderful if QNAP were to maintain a list of publicly exposed instances of their always buggy systems. At the same time, our listener brings up actually the most important point, which is, okay, what then? How do they contact the owners of the system that's publicly exposed? Bad guys don't need to contact anyone since they want to attack the systems. But good guys do need to contact someone since they want to remediate the trouble.

The good news is our VMware listener said: "It's been talked about quite a bit in the past in various contexts, but it's rather difficult at times to figure out who to contact if a vulnerable host is found." So I guess the good news is the idea is sort of in the air. But he's also correct about the privacy implications, which follow from any attempt to somehow make endpoints identifiable to everyone on the public Internet. Right? Because they would have to be everybody or no one. So we're left with the conundrum which is created by the asymmetry of the fact that bad guys want to attack exposed systems, while good guys only want to inform them. And I am thankful that our VMware listener spoke up. Thank you. And I don't think there's a good answer right now. But the way things are going, it seems like we need one.

Brad Jones said: "Steve, I know we're all tired of talking about the security of our passwords, but I'm interested in your thoughts on the following. Selecting a known Bitwarden master password that is already considered somewhat secure, then running it through something like a Base64 calculator to generate an actual password used to unlock the vault. As you would likely never remember the password, anytime you need to unlock you would run the known password through the calculator to generate the vault password. For example, although clearly you would never use Password12345678, that would generate the password you would use as your master password which is" - and then he has the Base64 conversion - UGFzc3dvcmQx and blah blah blah blah. You know, gibberish. He says: "There are many desktop and mobile applications that can complete this calculation without running the known password through an online encoder or decoder."

Okay. So algorithmic password generators are an interesting idea. The concept was to use an HMAC function which is essentially a keyed hash function. So the user would generate a single secret permanent key which would key the hash function. They would then enter, for example, the domain they're visiting into the hash function, and it would output an absolutely maximum entropy password. Since every domain would produce a different unique password, there would be no password reuse, and there would also be nothing to remember per domain since any domain's password could be recreated on demand from the domain name itself.

And actually this was part of the - this was the germ of the idea of how SQRL converts domain names into private keys which are then turned into public keys. And even before that, this idea intrigued me so much that I wondered whether it might be possible to create a truly secure paper-based encryption system. Our long-time listeners may recall that the idea I hit upon and then developed was to traverse a per-user customized Latin Square. I called the system "Off The Grid" since the passwords came from a grid, and the system was offline and used no electricity.

So anyway, Brad's idea is right, to suggest yet another means of generating unique per-site passwords that could be deployed using an algorithmic system based on something that could be input. And if you used a secret key and a hash function, you could just use the domain as a place to start.

Okay. Leo, our final break, and then I am going to introduce our listeners to something very new and very cool, Ascon, which has just received NIST standards endorsement.

Leo: Ooh, okay.

Steve: It is a new NIST standard for cryptography.

Leo: All right. Careful what you're asking for. You might get it.

Steve: Ask not what you can do for your crypto.

Leo: Your country. All right. I'm ready to - I'm asking for a friend. What is Ascon?

Steve: Okay. So last Wednesday the U.S. National Institute of Standards and Technology, you know, our NIST, announced that a family of authenticated encryption and hashing algorithms known collectively as Ascon, A-S-C-O-N, will be standardized for applications in lightweight cryptography. And lightweight cryptography does not mean less secure, as we'll see. So I'll have a little bit more to say about that in a second.

This final selection was the result of a four-year competition which ran from 2019 through just now, 2023, among competing proposals. This was the same process that led us to the selection of the Rijndael cipher to become the official AES standard. So that's, you know, Rijndael is military strength, if you'll pardon the expression. Ascon has now become the standard for providing high 128-bit security for lower end devices. And for example, there's an entire suite of symmetric cryptographic Ascon functions for the Arduino already exists.

So NIST summarized the competitive selection process by writing: "NIST has initiated a process to solicit, evaluate, and standardize lightweight cryptographic algorithms that are suitable for use in constrained environments, where the performance of current NIST cryptographic standards is not acceptable. In August 2018, NIST published a call for algorithms to be considered for lightweight cryptographic standards with authenticated encryption with associated data. That's Authenticated Encryption Associated Data, AEAD - we'll get to more of that in a second - and optional hashing functionalities. The deadline for submitting algorithms has passed. NIST received 57 submissions to be considered for standardization. After the initial review of the submissions, 56 were selected as first-round candidates. Of the 56 first-round candidates, 32 were selected to advance to Round 2." So they were whittling them down.

Okay. So what exactly does "lightweight crypto" mean? Rather than aiming for the highest conceivably needed security which would be adequate to protect data for the next several decades at least, NIST's Lightweight Cryptography competition was set up to select the contemporary crypto system that would be best suited for deployment on today's much more limited IoT systems. This means that, for example, while sharing the same 128-bit block length as the Rijndael AES cipher, because a cipher's block length really cannot be reduced much below 128 bits without compromising security, the key lengths can safely be reduced by half to 128 bits. This allowed the use of a family of faster, more efficient, and easier to implement algorithms which can run efficiently on much lighter weight hardware.

Okay, now, some have wondered whether dropping the key length in half to 128 bits is sufficient for contemporary security. The fact is, it's still insanely plenty of security. We only went to 256-bit keys because why not, if we have desktop and server machines and there's virtually, on those platforms, virtually no cost for doubling the key length. On the other hand, those are expensive algorithms, which is why Intel has now specific AES instructions because some of the things that AES requires the processor to do are extremely time-consuming. So Intel said, okay, let's put some instructions to speed up AES into our microcode. And they've been there now for a while. Well, if AES weren't hard to do, Intel would have never done cipher-specific instructions. Ascon doesn't need any of that, as an example. So the point is we did 256 bits because we could.

So let's examine the implications first of 128 bits. 2^128, right, the number of combinations, the number of possible keys, that's 3.4 x 10^38. So that's 34 followed by 37 zeroes. Okay, now, say that we want to brute force the key's value. And for the sake of argument let's say that it would be possible - it's not. But say that it would be possible to completely test a candidate key at the full clock rate, say 3.4 GHz, of a GPU. It's not possible to actually do that since a GPU, fast as they are, still requires many clock cycles to get any work done. But for the sake of this thought experiment, say that it was possible to fully test one 256-bit key every clock cycle at 3.4 gigatests per second. Okay, so that's 3.4 x 10^9, right, 3.4 GHz, gigatests per second. Brute force tests per second. Each clock cycle we're doing a brute force test. We can't, but for the argument let's say we could.

To test all possible 128-bit keys, where there are 3.4 x 10^38 possible keys, we divide 3.4 x 10^38 by 3.4 gigatests per second, that's 3.4 x 10^9. So we just subtract 9 from 38, yielding 29. This tells us that testing candidate keys, 128-bit candidate keys at the rate of 3.4 billion tests per second, which no actual system is even capable of coming close to doing, we would need 10^29 seconds to brute force all combinations.

Okay. Now, there are only 31.5 million seconds per year. If we triple that, it brings us to 94.5 million, which is close to 100 million. So we'll round that up to 10^8, which is 100 million. So 8 from 29 leaves 21. Thus brute forcing a 128-bit secret key at 3.4 billion guesses per second, which is not possible, would require roughly 3 x 10^21 years. Given that we all generally live less than 10^2 years, we're left with plenty of security margin.

In other words, 128-bit security, although we're so used to talking about thousands of bits for asymmetric key strength and 256 bits or more for AES and Rijndael, 128-bit security is an insanely ample amount of security. And now, thanks to this device-constrained cryptographic competition, we have this super-strong security available to lightweight embedded processors which don't need any special instructions because it's all simple.

NIST's original call for submissions explained their target, their need as follows. They said: "The deployment of small computing devices such as RFID tags, industrial controllers, sensor nodes, and smart cards is becoming much more common. The shift from desktop computers to small devices brings a wide range of new security and privacy concerns. In many conventional cryptographic standards, the tradeoff between security, performance, and resource requirements was optimized for desktop and server environments, and this makes them difficult or impossible to implement in resource-constrained devices. When they can be implemented, their performance may not be acceptable.

"Lightweight cryptography is a subfield of cryptography that aims to provide solutions tailored for resource-constrained devices. There's been a significant amount of work done by the academic community related to lightweight cryptography. This includes efficient implementations of conventional cryptography standards, and the design and analysis of new lightweight primitives and protocols."

Okay. So the winner was Ascon, a family of authenticated encryption and hashing algorithms which are not only designed to be lightweight and easy to implement, but also with added countermeasures against side-channel attacks. And Ascon was not only selected to be NIST's new lightweight crypto standard, but it also previously won the same spot in the CAESAR competition which ran for five years from 2014 through 2019. CAESAR (C-A-E-S-A-R) is the acronym for "Competition for Authenticated Encryption: Security, Applicability, and Robustness." And yes, that's clearly one of those reverse-engineered acronyms that's always a bit awkward. So Ascon is also CAESAR's primary choice for lightweight authenticated encryption.

And of course during the NIST competition it was already obvious that Ascon had won the CAESAR competition since that closed in 2019. So really the question was is there anything more that we know now, subsequent to the CAESAR competition, that is better than Ascon. And the answer? No. There were more than 50 submissions of other things. And Ascon was it. And I've mentioned authenticated encryption here. We've talked about the need in the past for both encrypting the data for privacy and also authenticating the data to prevent its manipulation at any time. And assuming that the two steps of encryption and authentication are separate, as they were originally, we've examined the question of in which order the two operations should be accomplished.

The universal agreement is that you first encrypt, and then you apply authentication to the encrypted result. The reason is, on the receiving decrypting end, the message is first authenticated to assure it has not been tampered with, not modified in any way, and only if it authenticates should the message then be decrypted. This prevents the use of some very subtle attacks that we discussed years ago.

But the use of so-called "authenticated encryption" offers a better solution. As its name implies, authenticated encryption algorithms are able to perform both operations at the same time. And there's one very valuable extension to this known as Authenticated Encryption and Associated Data, or AEAD for short. AEAD algorithms allow the so-called "associated data" to remain unencrypted and in the clear while still being encapsulated within the authentication envelope. So you can see it, but you can't change it.

I have some experience using these algorithms, since SQRL's local on-disk identity storage format needed to have some local storage which would always be left, well, some that would be kept encrypted on the disk, but also some other that, for example, the user's settings, which would need to be available all the time even without the user's key. And the whole thing, the entire package needed to be tamper proof. So I chose to use the AES-GCM cipher, which is an AEAD cipher. This perfectly resolved the need for having both secret and non-secret data that could not be altered without breaking the authentication of the whole.

So in any event, what we're getting is in this lightweight competition is an AEAD cipher which is going to be extremely useful. It's what NIST wanted, and it's the contest-winning technology that Ascon provides to IoT devices. It also brings and provides similar features to hashing because Ascon did also offer a hashing technology and actually a lot more. It was designed by a team of cryptographers from the Graz University of Technology, Infineon Technologies, Lamarr Security Research, and Radboud University.

So its bullet points, that is, of its benefits are authenticated encryption and hashing with fixed or variable output length with a single lightweight permutation. And I'll be talking about this permutation in a second. Provably secure mode with keyed finalization for additional robustness. Easy to implement in hardware and software. Lightweight for constrained devices. It's got small state, meaning not much storage needed, a simple permutation box, and robust mode. Fast in hardware. Fast in software.

Pipelinable, bit-sliced, it has a bit-sliced 5-bit S-box for 64-bit architectures. Scalable for more conservative security or higher throughput. Timing resistance; no table look-ups or additions. Side-channel resistance; the S-box is optimized for countermeasures. The key size equals the tag size equals the security level, that is, they're all 128 bits. Minimal overhead; the ciphertext length equals the plaintext length so there's no padding needed. Single-pass, online encryption and decryption, and it uses a nonce.

So the NIST requirements were for a key size no shorter than 128 bits, and there could be a family of variants having different parameter lengths. The design targets were interesting. NIST wrote: "Submitted AEAD algorithms and optional hash function algorithms should perform significantly better in constrained environments" - meaning hardware and embedded software platforms - "compared to current NIST standards. They should be optimized to be efficient for short messages, as short as 8 bytes. Compact hardware implementations and embedded software implementations with low RAM and ROM usage should be possible.

"The performance on ASIC (Application Specific Integrated Circuits) and FPGA should consider a wide range of standard cell libraries. The algorithms should be flexible to support various implementation strategies - low energy, low power, low latency. The performance on microcontrollers should consider a wide range of 8-bit, 16-bit, and 32-bit microcontroller architectures. For algorithms that have a key, the preprocessing of a key in terms of computation time and memory footprint should be efficient.

"The implementations of the AEAD algorithms and the optional hash function algorithms should lend themselves to countermeasures against various side-channel attacks, including timing attacks, simple and differential power analysis, and simple and differential electromagnetic analysis. Designs may make tradeoffs between various performance requirements. A submission is allowed to prioritize certain performance requirements over others. To satisfy the stringent limitations of some constrained environments, it may not be possible to meet all performance requirements stated in the previous paragraph. The submission document should, however, explain the bottlenecks that were identified and the tradeoffs that were made." And basically no tradeoffs were necessary. Ascon does it all.

So we now have a new family of power, memory, and general resource-efficient algorithms that have been studied and pounded on, now for years, by many academics and cryptographers. GitHub offers implementations in C, C with some assembly, Python, Java, and RUST. And there are RISC-V implementations, as well. I have links to everything in the show notes.

GitHub also has an Ascon Suite of cryptographic functions all based on what's known as, and I mentioned this before, the Ascon permutation. The permutation is the core cryptographic function at the heart of Ascon, highly efficient, super secure, now proven. That GitHub page provides the open source code for the following functions. We've got, of course, Authenticated Encryption with Associated Data. We've got hashing. We've got a pseudorandom function, a Message Authentication Code, an HMAC-based Key Derivation Function, Hashed Message Authentication Code (HMAC). We have AEAD with side-channel protections. We've got a Keyed Message Authentication Code (KMAC), Password-Based Key Derivation Functions (PBKDF), a pseudorandom number generator, synthetic initialization vector, extensible output functions, and direct access to the ASCON permutation.

All of those are available through the API of the code here. It's GitHub.com/rweather/ascon-suite. And there's a related page with all of the same functions for the Arduino. So the industry, as I said, has a robust and seriously secure set of lightweight functions, which have already been ported to many languages, platforms, and hardware, which are tuned for and suited for the needs of embedded resource-constrained IoT-style devices. And I obviously already made the case that 128-bit key length, that is just fine. That'll last us our lifetime and, well, what, 10^18 times more.

Leo: So why not just replace AES? Why not replace Rijndael with this?

Steve: You really could. I mean, I think they went high. Rijndael is available in 256-bit, 384-bit, and 512-bit key lengths. So you could just go insane with Rijndael. I agree.

Leo: Do you need to, I guess is the question. And even if you went to 512, would it survive a quantum computer?

Steve: Well, this is all symmetric.

Leo: Oh, it doesn't matter.

Steve: And quantum computers do not attack symmetric ciphers at all.

Leo: Right, right, okay.

Steve: Yeah. So if I have any need in the future, I'll be using these because Ascon, it's been really picked over now for many years. And you can imagine that the 50 other submissions, they were trying to find a problem with Ascon in order to have theirs chosen, but they couldn't.

Leo: I think, you know, this would be good for PGP to replace, I think they're using Triple-DES for the symmetric key.

Steve: Yeah.

Leo: It would be a nice improvement. Not that speed is a big issue. But still, just something that's easier to implement maybe. It's interesting. Ascon. Huh.

Steve: We have it now.

Leo: We've got it. Thanks to rweather, whoever that is. Is rweather a person, or is it...

Steve: I didn't look to see. But basically you're taking the core innovation, which is this permutation box, which is well defined, and then you're building all these other functions around it. And we sort of know how to do that now.

Leo: I have a feeling rweather's a person who's just really good at this stuff.

Steve: Might be. He may have just done an implementation of these functions.

Leo: Yeah, yeah. Interesting. Wow. You know what, this is great. You learn so much listening to this show. And you stay up to date. And I know that's why so many people in cybersecurity and IT listen to Security Now!, and we're glad you do. I hope you'll keep listening.