Global Privacy Control

Description: This week we're going to examine the success of the abbreviation overloaded DoD's DIB-VDP pilot program. We're going to introduce the relatively new OpenSSF - Open Source Security Foundation - and its Package Analysis Project. We're going to look at some hopeful new privacy legislation recently passed in Connecticut's house which if signed into law would cause it to join four other privacy-progressive states, and we're going to look at Moxie Marlinspike's irreverent rationale for the need for port knocking. Then, after sharing some interesting listener feedback, we're going to look at the background, implementation and future of a very encouraging development in user web browser and Internet privacy.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-869.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-869-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with an acronym soup, including DoD's new DIB-VDP program. We're going to talk about OpenSSF, and then a simple way to tell the world you don't want to be tracked. And this time it really works. You can stay tuned, I hope, for this one. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 869, recorded Tuesday, May 3rd, 2022: Global Privacy Control.

It's time for Security Now!, the show every Tuesday that answers the vital question, what the heck? And here's the man with the answers to all your questions, Steve Gibson. What could possibly go wrong is actually the...

Steve Gibson: What could possibly, yeah, that really is our tag line.

Leo: That's the question you really, yeah, you really do answer. Steve from GRC.com. Hello, Steve.

Steve: So I have sort of a blast from the past, some good news. This is Security Now! Episode 869 for this is our first episode of May 2022 titled "Global Privacy Control." And the subtitle is "Hope Springs Eternal." I'm hopeful. That's my nature. I've always been hopeful. I refuse not to be hopeful. Why not have hope? Anyway, rather than a ton of little topics, the way it turned out, we have a number of bigger topics. We're going to examine the success of the abbreviation-overloaded DoD's DIB-VDP pilot program. Boy, the government loves their initials.

We're going to introduce the relatively new OpenSSF, the Open Source Security Foundation, with the introduction of their first project, which they call the Package Analysis Project. We're going to look at some hopeful new privacy legislation recently passed in Connecticut's house which, if signed into law, would allow it to join four other existing privacy-progressive states, among which California and Colorado number. And now they're being called - they're being considered the Three C's: Connecticut, California, and Colorado. And we're going to talk about that.

We're also going to look - I got a kick out of this. Someone must have tweeted this to me. Moxie Marlinspike, who, of course, famous for being the lead designer of the Signal messaging platform, he developed his own port knocking solution a decade ago. And I just, I loved - he asked himself at the end of his page on GitHub about this, like why do you even need this? And I know that everyone will get a kick out of his answer. We've also got a bunch of listener feedback. And then we're going to talk about what Global Privacy Control is and why this time might be different. So I think another great podcast for our listeners.

Leo: Excellent. Very excited. I didn't even ask, Steve. Do we have a Picture of the Week?

Steve: We do, as a matter of fact.

Leo: Oh, good.

Steve: It'll be a bit of a surprise.

Leo: All right.

Steve: One of our listeners posted this in the SpinRite development group, but I just got a kick out of it and thought...

Leo: Oh, it's really good news.

Steve: It's SpinRite running on a MacBook.

Leo: Wow. Wow. Look at this. Holy cow.

Steve: Yeah. And the other cool thing is I remember a long time ago guessing that I thought that the new SpinRite would be so fast that it could do - I think I had said half a terabyte per hour was my estimate of its speed. As we can see from the screen here, and he's just run the benchmark where we're benchmarking the front, the middle, and the end of the drive, and we can see that this 250GB drive in the MacBook would take 7.9 minutes for SpinRite to scan. So four times that is about 30. So that would mean 30 minutes to do a terabyte. In other words, we're doing two terabytes per hour rather than half a terabyte per hour. So it really becomes far more feasible to run SpinRite on today's really massive drives.

Leo: How are you running on a MacBook? Do you have to boot into FreeDOS or...

Steve: You do need to have Boot Camp.

Leo: Ah.

Steve: So it uses Boot Camp.

Leo: Ah, okay. So not the modern MacBooks, unfortunately.

Steve: Exactly. Not the most recent MacBooks. It did remove Boot Camp. And you can see in the picture he's got a thumb drive plugged into the USB there in the back right of the keyboard. And so he intercepted the boot and booted FreeDOS. So Intel only, and you do need Boot Camp. So certainly on all of the older MacBooks it'll run perfectly. So anyway, just a very cool little picture. Caught me by surprise. I said, hey, cool, because I've actually run it on my MacBook years ago when I diddled the keyboard. The reason it wouldn't ever run before was that once SpinRite starts going, it directly polls the keyboard hardware.

Well, PCs have keyboard hardware. Macs don't. They have a USB keyboard which interfaces and sort of emulates the BIOS. But SpinRite wasn't - it was like pushing past the emulation so you couldn't run it, even though everything else worked about it, it just wouldn't, you know, the keyboard died after you started up SpinRite. So I've made sure that won't happen in the future. And it's, wow, it's going to be so much faster.

Okay. The DoD's DIB-VDP Pilot...

Leo: M-O-U-S-E, yes.

Steve: Yeah. So the program of the DoD - that's our U.S. Department of Defense - was named the "DIB-VDP Pilot Program." And despite suffering from program name abbreviation overload, it just successfully finished its year-long pilot test. The DIB stands for Defense Industrial Base. VDP is Vulnerability Disclosure Program, thus DIB-VDP. It was a 12-month voluntary event established collaboratively by DC3's - oh, that's the - no. Oh, DC3's, no, that's DCSA. Anyway, it's here somewhere. Oh, Defense Cyber Crime Center is the DC3. Oh, my god.

So established collaboratively by DC3's, that's the Defense Cyber Crime Center, DoD Defense Industrial Base Collaborative Information Sharing Environment, and I kid you not, that's the DCISE. So, oh, and the DoD Vulnerability Disclosure Program, thus the DoD VDP, and the Defense Counterintelligence and Security Agency, the DCSA. Right.

That year-long bug bounty program scrutinized a fraction, and this is key because we're going to see what a fraction it was, a fraction of the U.S.'s massive defense industrial base. Yesterday, on Monday, the effort's organizers announced that more than 400 valid vulnerabilities had been uncovered.

So during the 12-month pilot program, nearly 300 vulnerability security researchers from HackerOne - that's where they were sourced because you had to be vetted. And we talked about this a year ago, that this was going to start, and you had to be vetted, you

just couldn't come wandering in and attack the U.S. government and hope to get away with it. So HackerOne was the coordinator. Those nearly 300 vulnerability researchers filed 1,015 reports during their examination of the networks of individual participating defense contractors to the DoD. As a result of their scrutiny, those 401 vulnerabilities were deemed actionable and required remediation.

The pilot effort was run in coordination with the, oh, my god, the Defense Cyber Crime Center, that's the DC3; and the Defense Counterintelligence and Security Agency, the DCSA. The program began with 14 companies and 141 public-facing assets. That is to say, those were the targets. Fourteen companies with 141 public-facing assets were initially offered to these HackerOne hackers. And over the course of the year that effort was expanded to include 41 entities and 348 systems.

Now, the point of this is, this is minuscule. What should give the program managers pause is that 401 actionable vulnerabilities were found in those 348 systems belonging to 41 entities, yet somewhere between 100,000 and 300,000 companies contract directly with the Pentagon and its components. So like I said, you know, literally, so more than actionable vulnerability found per system on average, and about 10 per entity that contributed and participated voluntarily in the program. And we have a minimum, and I'm not sure why the number that was quoted was so large, I mean, to say between 100,000 and 300,000 is like, okay, well, how can it be a three-to-one, give or take? But anyway, that's the U.S. government for you. So this legitimizes the long-running worry - which, yeah - about the latent digital vulnerabilities of the firms that make up the Department of Defense's supply chain, which has, as we know, been rocked by a number of major breaches over the years.

For example, one of the most notorious incidents, which I referred to sort of in passing last week, occurred back in 2009 when suspected Chinese hackers broke into one of the contractors working on the F-35 Joint Strike Fighter, which as we know is the most expensive weapons platform ever in U.S. history, and made off with its design data. Whoops. So Melissa Vice, the interim director of the DoD Vulnerability Disclosure Program, so she was directing this effort, said: "The pilot bug bounty, which concluded at the end of April, intended to identify whether similar critical and high-severity vulnerabilities existed on small to medium cleared and non-cleared DIB - that's Defense Industrial Base, so that's collectively how these companies, these contractors are known - company assets with potential risks for critical infrastructure and the U.S. supply chain." That was Melissa saying that.

So armed with this data, an analysis of the DIB vulnerability report management network, which is yet another abbreviation, will now take place in order to document the pilot program's lessons learned and inform the way forward for a fully funded program. Alex Rice, the co-founder and CTO of HackerOne, who as I said was the organizing, you know, HackerOne provided the hackers, he said: "With CISA now mandating vulnerability disclosure for government agencies and federal contractors, the DIB-VDP takes the practice a leap forward by demonstrating the efficacy of Vulnerability Disclosure Programs in the real world."

So the fiscal 2022 defense policy bill required the DoD to assess the feasibility of allowing a threat-hunting program on the defense industrial base to weed out foreign hackers and study the possibility of establishing a threat information sharing program for what has become a sprawling DoD enterprise. Yeah, if you've got 300,000, well, 100,000 or 300,000, you're not really sure, you know, subcontractors, that's sprawling. Earlier this year the DoD's CIO, John Sherman, said the Pentagon was "getting rolling on both of those" examinations, which, as I said, the fiscal 2022 defense policy bill required that they do this, so they did. Now we've seen, and the bureaucracy has witnessed, the clear feasibility of these programs.

And when you stop to think about it, this approach makes so much sense. No sane and talented white hat hacker is going to poke at government networks and systems at the risk of triggering a law enforcement action against them. Right? We've seen exactly that happen in the past. So that must be done within the bounds of a sanctioned bug hunting program. And similarly, no sane and talented white hat hacker, no matter how patriotic, is going to invest their valuable time doing such poking for free, when they could be potentially earning a bounty from poking at some private enterprise's networks and systems that pays a bounty.

So the government needs to pay. And no private enterprise has as much money lying around as the government, or a government, so paying is not a problem. So it really ought to be that fully sanctioned government bug bounty programs which allow hackers to earn bounties which are competitive with those in the private sector become a permanent reality in the future. Successes such as we're seeing here, and what we saw during this DoD DIB-VDP though they really do need a sexier name for it move us another step in that direction.

So this was all good. And a lot of people said, wait a minute, 10 vulnerabilities which needed action per contractor on average? And we have how many hundreds of thousands of contractors? So, yeah.

Okay. We've never talked about the relatively recently formed OpenSSF. Hard to pronounce, or say. OpenSSF stands for Open Source Security Foundation. It's a Linux Foundation project created in 2020 by a very large group of significant tech and financial firms with the charter to help steer, guide, and share open source security tools. Which unfortunately are lacking.

Its membership is divided into two classes, Premier and General members, with the premier members in alphabetical order being 1Password, AWS, Atlassian, Cisco, Citi, Coinbase, Dell, Ericsson, Fidelity, GitHub, Google, Huawei, Intel, IBM, JFrog, JP Morgan, Meta, Microsoft, Morgan Stanley, Oracle, Red Hat, Snyk, Sonatype, VMware, and Wipro. And there are about three times as many general members, which I won't enumerate. But many are equally notable, like there's Bloomberg, Comcast, Goldman Sachs, MongoDB, Spotify, SUSE, Tenable, Tencent, Carnegie Mellon, and Mitre. So again, this is a heavyweight group. I also saw in some reporting that Samsung was a member, but they weren't listed on the group's official page at openssf.org, by the way.

So that page, openssf.org, says: "Open source software is pervasive in data centers, consumer devices, and applications. Securing open source supply chains requires a combination of automated tooling, best practices, education, and collaboration." And so they divide the effort that they're organizing into three components. There's Working Groups, where they say: "Collaborate on the planning, design, and delivery of security tooling and best practices that secure critical open source projects." Then they have Town Halls: "Stay informed about the latest happenings in open source security and engage with experts in our community." And Training: "Take free courses on secure coding practices as part of our Software Development Fundamentals Professional Certificate."

So all of that is nothing but good. Again, it feels like it's coming along a little late in the game, but wow. Better than never. Okay. So that's the OpenSSF. We're talking about this previously undiscussed group today because last Thursday they announced the first results, which were a success, with some caveats that I'll discuss, of their project known as the Package Analysis Project. I've got a link in the show notes for anyone who's interested.

To introduce it, they explain package analysis: Scanning open source packages for malicious behavior. And so they said: "Today we're pleased to announce the initial

prototype version of the Package Analysis project, an OpenSSF project addressing the challenge of identifying malicious packages in popular open source repositories." And talk about something we've been talking about a lot recently, open source repositories. They said: "In just one month of analysis, the project identified more than 200 malicious packages uploaded to PyPI and npm."

Okay, so think about that for a minute. Thanks to this project, 200 previously present but unknown malicious Python and JavaScript packages were found. In a minute I'll share some of the details about what they found. But their announcement continues, saying: "The Package Analysis project seeks to understand the behavior and capabilities of packages available on open source repositories: what files do they access, what addresses do they connect to, and what commands do they run? The project also tracks changes in how packages behave over time, to identify when previously safe software begins acting suspiciously. This effort is meant to improve the security of open source software by detecting malicious behavior, informing consumers selecting packages, and providing researchers with data about the ecosystem. Though the project has been in development for a while, it has only recently become useful following extensive modifications based on initial experiences."

Okay. So what they've built is a big heuristic engine which is capable of performing static code analysis, currently of packages residing in Python and JavaScript repositories. And, as is the case with heuristics, which we can think of as complex rules of thumb, tweaking and adjusting of weighting factors is typically necessary. But if the signals they're able to obtain from their static code analysis are strong enough, robust determinations should be possible.

So they continue: "The vast majority of the malicious packages we detected are dependency confusion and typosquatting attacks. The packages we found," they said, "usually contain a simple script that runs during an install and calls home with a few details about the host. These packages," they conjecture, "are most likely the work of security researchers looking for bug bounties, since most are not exfiltrating meaningful data except the name of the machine or a username, and they make no attempt to disguise their behavior. Still, any one of these packages could have done far more to hurt the unfortunate victims who installed them, so Package Analysis provides a countermeasure to these kinds of attacks."

They said: "There are lots of opportunities for involvement with this project." So they're also soliciting from the community, and our listeners are part of that community. They said: "And we welcome anyone interested in contributing to the future goals of" - there are four of them - "detecting differences in package behavior over time, automating the processing of the Package Analysis results, storing the packages themselves as they're processed for long-term analysis, and improving the reliability of the pipeline."

They then finished: "Check out our GitHub Project and Milestones for more opportunities, and feel free to get involved on the OpenSSF Slack. This project is one of the efforts of the OpenSSF Securing Critical Projects Working Group. You can also explore other OpenSSF projects like SLSA" - I didn't dig into what that was - "and Sigstore, which expand beyond the security of packages themselves to address package integrity across the supply chain."

So this is very welcome news. As we know, for the past several weeks especially I've been lamenting the inherent lack of security of the systems that have been built to make sharing code libraries virtually effortless. The trouble has been that polluting these code repositories is also effortless. And recently we've seen this process even being automated with bots to further thwart code identification and its takedown.

I mentioned some specific examples from among the more than 200 malicious packages discovered during the first month of the system's use. The first two attacks listed in their case study published on GitHub were against Discord clients and their users. They explained that Discord attacks are focused on attacking Discord accounts or clients. They said stolen accounts can be resold, used for fraud like cryptocurrency scams, or used to gain insider knowledge or secrets.

The first one that was over on PyPI is discordcmd. And in their analysis results they said: "This Python package will attack the desktop client for Discord on Windows. It was found by spotting the unusual requests to raw.githubusercontent.com, Discord API, and ipinfo.io." They said: "First, it downloaded a backdoor from GitHub and installed it into the Discord electron client. Next, it looked through various local databases for the user's Discord token. It grabbed the data associated with the token from the Discord API and exfiltrated it back to a Discord server controlled by the attacker." And what's so cool, of course, is that they found this with automated scanning heuristics.

The second Discord malware was over on npm, and it was titled "colorsss." In their analysis result they said: "Similar to discordcmd above, this npm package attempts to steal a Windows user's Discord account token and was discovered by identifying calls to the Discord API. This package first searches through local browser databases for a token, queries the Discord server to discover details about the token, and exfiltrates the details to a Discord server controlled by the attacker."

They also during this first month of scanning discovered a couple of remote shells. They explain, for those who don't know: "A remote shell is used by an attacker to provide access to a command shell running on a target machine over the network. They're usually 'reverse shells' that connect back to an attacker-controlled machine."

So we have over on npm, this one was roku-web-core/ajax. And they explained: "During install, this npm package exfiltrates details of the machine it's running on and then opens a reverse shell, allowing the remote execution of commands. This package was discovered from its requests to an attacker-controlled address."

Another one in Python, over on PyPI, is secrevthree. And the analysis says: "This package opens a simple reverse shell when a specific module is imported from the library, allowing remote command execution. It uses a simple obfuscation technique to make it harder for a reader to notice. This package was discovered from the request to the attacker-controlled address."

And lastly, over on npm, this one was random-voucher-code-generator. And they said: "When the library is imported, it queries a server running on their Heroku cloud platform. The server response includes a command and a flag indicating whether or not the command should be run. In the response we observed, the command stopped a process managed by 'PM2.' However," they said, "this same response can be changed to run any command the attacker wished to execute, including opening a reverse shell. The library then uses voucher-code-generator to provide the advertised functionality." So, you know, you get your voucher-code-generator. But you also get a reverse shell that you probably didn't count on. They said: "This package was discovered from the unusual request to the Heroku server."

Their case studies page finishes by talking about dependency confusion and typosquatting, you know, where they just, you know, we've been talking about that recently where remember the dependency confusion. They did note that packages typically used a high version number, so the package managers that don't know better see that a high version number is available, higher than the one they have, and say, oh, I'd better go get that. And of course when they do they get themselves pwned. So, not good.

As we've covered previously - oh, and I was just going to talk about exactly that, the use of high version numbers in order to trick package managers into obtaining it.

Google, who is an active participant in this effort, added a bit of additional background in their own blog posting. They explained, they said: "Despite open source software's essential role in all software built today, it's far too easy for bad actors to circulate malicious packages that attack the systems and users running that software. Unlike mobile app stores that can scan for and reject malicious contributions" - and this Google speaking, we know how tough even that is for them - "package repositories," they write, "have limited resources to review the thousands of daily updates and must maintain an open model where anyone can freely contribute." Thus one of the big problems. "As a result, malicious packages like ua-parser-js" - which we actually had spoken of previously - "and node-ipc are regularly uploaded to popular repositories despite their best efforts, with sometimes devastating consequences for users.

"Google," they write, "a member of the Open Source Security Foundation, is proud to support the OpenSSF's Package Analysis project, which is a welcome step toward helping secure the open source packages we all depend upon. The Package Analysis program performs dynamic analysis of all packages uploaded to popular open source repositories and catalogs the results in a BigQuery table. By detecting malicious activities and alerting consumers to suspicious behavior before they select packages, this program contributes to a more secure software supply chain and greater trust in open source software. The program also gives insight into the types of malicious packages that are most common at any given time, which can guide decisions about how to better protect the ecosystem."

Okay. So this is all for the good. The bad news is that this again puts us back into, I think, a pretty clear cat-and-mouse game. Because the project is inherently open, the bad guys can learn what the system is keying on and simply arrange to avoid it. How many times in those descriptions I just shared did we hear: "This package was discovered from the request to the attacker controlled address." Right? Or another of those, several of those descriptions said: "These dependency confusion attacks were discovered through the domains they used, such as burpcollaborator.net, pipedream.com, interact.sh, which are commonly used for reporting such attacks."

So the bad news is, once the bad guys learn that these simple factors, which did not previously interfere with their nefarious aims, have become problematic for them to use, they'll simply stop using such obvious targets by arranging to use non-obvious addresses, such as by proxying their connections through any of the gazillion compromised consumer routers on the Internet. We've already discussed such readily available proxying being used to hide connections back to command-and-control servers. In this case, it wasn't necessary to do that before. Now it is, so it'll happen.

We already saw something similar with the adoption of package submission bots, which we were just talking about the other day, being used to create unique per-package-submission accounts. This prevented casual linkage to other malware which had also been submitted by the same malefactor. The underlying problem is that, at the moment, and as Google referred to, saying it was crucial and necessary, anyone is able to offer packages for submission. There's no reputation system in place. And the problem with reputation-based filtering is that, yes, it can be discriminatory, preventing valid unknown publishers who've not yet established a reputation from contributing their efforts.

So we still have some problems to solve. Nevertheless, it is wonderful that the inherent value and power of open source software and its need for some form of security oversight is being formally recognized; and that we now have an OpenSSF, an Open Source Security Foundation which will focus and work towards introducing improvements in this valuable ecosystem. And again, if anybody is looking for something to contribute to, some way to apply their skills, I think this is great because the gotcha here in the

whole OSS movement is that security is as big a problem there as it is for the closed source world.

Leo: Hydration completion?

Steve: Yes.

Leo: Yes.

Steve: Thank you. So Connecticut, a state in the U.S. for those who are...

Leo: I've heard of them, yeah.

Steve: ...not in the U.S. It's difficult to spell, but once you get the trick it's okay. The U.S. federal government is not moving forward on consumer privacy protections, so several states have taken matters into their own hands.

The state of Connecticut's General Assembly advanced a bill last Thursday that would offer residents a useful set of baseline privacy rights. If signed into law by Connecticut's Governor Ned Lamont, Connecticut would join four other states which have already enacted similar legislation. Those four are California, Colorado, Virginia, and Utah.

Connecticut's bill, which is their SB 6, which is named

"Act Concerning Personal Data Privacy and Online Monitoring." If signed, and the governor's spokespeople are not committing but are not saying no, and it's expected to go, it's expected to pass, it would take effect July 1st of next year, 2023. So not immediately even when signed. And it closely resembles the privacy laws which have already been passed in Colorado, Virginia, and Utah in that it allows residents of those states to opt out of sales, targeted advertising, and profiling. And after two years, by 2025, the law requires companies to acknowledge opt-out preference signals for targeted advertising and sales.

Now, websites and companies would be required to proactively obtain consent to process sensitive data, and need to offer Connecticut residents ways to revoke that consent. And the law provides that organizations will have no more than 15 days to stop processing data as soon as consent is revoked. Parental consent is needed for any website to collect personal data from children under the age of 13, but businesses are banned from collecting personal data and using targeted advertising on children between the ages of 13 and 16.

The bill forces companies to honor browser privacy signals like the Global Privacy Control, thus the title of this podcast, and we'll be talking about that here in a minute, so that consumers can opt out of data sales at all companies in a single step. And as we know, Global Privacy Control - oh, yeah, I have in my notes Global Privacy Control is today's main topic. So we'll be discussing it in detail shortly.

So Keir Lamont, the senior counsel at the Future of Privacy Forum, added that Connecticut's privacy bill goes beyond existing state privacy laws by "directly limiting the use of facial recognition technology, establishing default protections for adolescent data, and strengthening consumer choice, including through requiring recognition of many

global opt-out signals. Nevertheless," he says, "a federal privacy law remains necessary to ensure that all Americans are guaranteed strong baseline protections for the processing of their personal information."

Now, the law also joins California and Colorado in adding sunset clauses to the so-called "right to cure." In other words, placing an end to that right which has been regarded and treated as a "get out of jail free" card. This "right to cure" describes a process where companies are given a set amount of time to fix violations before enforcement action can be taken or lawsuits can be filed. In other words, it basically allows them to bypass the law. Its presence has been a hotly debated topic in many state assemblies across the country. It's been the reason why several previous privacy bills have failed to pass. Consumer Reports, which worked with Connecticut lawmakers on their bill, called the "right to cure" provisions in most privacy laws, they're the ones who coined the term, a "get out of jail free" card for companies violating consumer privacy.

However, Connecticut's right to cure provision sunsets on December 31st, 2024. Colorado's provision sunsets the next day on January 1st, 2025. But our, and I say "our," Leo, because you and I are both in California, California's sunsets the first of next year, January 1st, 2023, which puts teeth in this for California. Once these sunset, the states will be able to take enforcement action against organizations that violate the law.

California, Colorado, and Connecticut are therefore being referred to as the "3Cs" because they all have useful sunset provisions. A lawyer, David Stauss, who serves as chair for the Husch Blackwell law firm's Privacy and Cybersecurity Practice Group, said: "Through joint enforcement, the 3Cs of state privacy law will be in a unique position to dictate the future of U.S. privacy." Because as we know, for state laws, any business doing business in that state, which is to say all businesses, need to abide by the laws within that state. So he said: "Through joint enforcement, the 3Cs of state privacy law will be in a unique position to dictate the future of U.S. privacy, assuming the continuing absence of any overriding federal law. But this will not be the case with Virginia and Utah, where 'right to cure' will endure."

So Connecticut's SB 6 also establishes a privacy working group to analyze a number of issues and provide a report by September 1st of this year. And without federal support, it's all a bit of a sticky wicket. Several states have spent years attempting to pass their own privacy laws due to the lack of any movement on privacy legislation at the federal level. Calls for a federal privacy law have ramped up since the European Union's infamous GDPR, their General Data Protection Regulation, became enforceable in 2018, which has served as a model for similar laws in Japan, Brazil, South Korea, and elsewhere.

But New York, Texas, Washington, and dozens of other states have faced so far insurmountable pushback when attempting to move their own privacy legislation forward due to backlash from businesses that complain the bills will create a significant amount of extra work for effectively any business with a website. Oh, boo hoo. Right. You have to pay attention to a signal, an opt-out signal from the browser, and then behave accordingly. Gee.

Anyway, the 3Cs - California, Colorado, Connecticut - won't care and won't need to. Any web-based business with a customer in California, Colorado, and presumably Connecticut once this legislation passes, will need to get their digital act together soon in the case of California, as I said, by the end of this year or face legal challenges. As we'll see, the best news is that the adoption of the new Global Privacy Control browser signal means that we're not all going to be needing to keep clicking something like "I want my privacy!" at every website we visit, unlike that horrible cookie mess that we're facing.

Okay. Now, I ran across this, and I loved it. This is Moxie. We all know Moxie Marlinspike. We've spoken of him often. He's most famous for his hand in the development of Signal, which provides a secure technology for true end-to-end encryption of instant messages. We did a podcast on it years ago. If anyone is interested in how Signal works, of course WhatsApp is based on Signal. Unlike Telegram, Signal employs standard, proven, and well-understood cryptographic primitives. You can look at it and go, yeah, this is secure. There's no hocus pocus, mumbo jumbo there. It's just very clever. And that's where that ratchet concept was introduced in order to allow secure messaging even in an asynchronous message-passing environment.

Well, it turns out that 10 years ago Moxie took his own stab at his own design and implementation of port knocking. For anyone who's interested, it's on GitHub, and I've got the link in the show notes. And I won't go into it in depth. It's just another way to solve the same problem. But what I loved, and the reason I'm mentioning it today, is that Moxie ended his page with a section titled "Why Is This Even Necessary?" which I wanted to share because it'll sound quite familiar. In answer to that question, Moxie wrote: "You are running network services with security vulnerabilities in them." Period. And then he said: "Again, you are running network services with security vulnerabilities in them."

He said: "If you're running a server, this is almost universally true. Most software is complex. It changes rapidly, and innovation tends to make it more complex. It is going to be forever hopelessly insecure. Even projects like OpenSSH that were designed from the ground up with security in mind, where every single line of code is written with security as a top priority, where countless people audit the changes that are made every day, even projects like this have suffered from remotely exploitable vulnerabilities. If this is true, what hope do the network services that are written with different priorities have?"

He said: "The name of the game is to isolate, compartmentalize, and expose running services as little as possible." He says: "That's where knockknock comes in." That's the name of his solution. "Given that your network services are insecure, you want to expose as few of them to the world as possible." He finished, saying: "I offer knockknock as a possible solution to minimize exposure."

So from the mouth of Moxie, there it is. As I said, it should sound familiar to anybody who's tired of listening to me say the same thing on this podcast. All of the evidence that we have, that we cover every week, proves this point over and over and over, exactly as Moxie stated. So it's a reason why I'm a big fan, when you have fixed IPs on your endpoints, absolutely use IP-based firewall filters so that nobody anywhere else has any access to the ports which you have opened between two fixed locations. And even if they're mostly fixed, you know, as I've said, if you've got a cable modem, that IP is not changing often, like nearly never. So even there it's feasible to do IP-based point-to-point filtering. And if you need to connect to clients that are going to have a floating IP, some form of port knocking, some form of out-of-band or kind of quasi in-band authentication in order to then give that IP and port temporary access, just it makes all the sense in the world. And there are widely available tools for making it happen. You just have to decide to.

Okay. Some feedback from our listeners. Dstacer wrote, from @dstacer, he said: "How can a cryptocurrency wallet in a browser be a very bad idea, while LastPass and other browser-based password managers are a very good idea?" You know, that's a really good question I had to think about a bit, about how to explain why I still feel that both of those apparently opposing evaluations can both be true at the same time. I think it boils down to necessity and practicality. It is not necessary to use a browser as a cryptocurrency wallet. It would be far safer to perform a wallet's operations in a more secure container, in any other more secure container. And it would be, and is, practical to do so. Most people do.

I also agree that a browser is not as secure as I would wish it were for the storage of all of my most precious static passwords in any password manager. But it's pretty much necessary to integrate a password manager into our browser, and it would be far less practical not to have that integration. If a password manager were not integrated, we would need to manually look up every site we were visiting, then display, copy, and paste our username and password for that site into the browser. And it passes through the clipboard, which has been a problem in malware that monitors the contents of our clipboards. We would also be at an increased risk from lookalike phishing domains, since we might be fooled by "paypol.com," whereas an integrated password manager would not be.

This podcast has well established that browsers are today's number one attack targets. They are the attack surface which we all present to the Internet as we move about. They're not perfectly secure. They're not as secure as we could wish they were. Yet we are not seeing reports of browser native password databases, nor third-party password manager extensions, being hacked. So it must be that, unlike the mistake we discussed last week that was made in the browser-based cryptocurrency wallet add-on, those who are authoring and maintaining both browser-native and third-party password managers are really taking their jobs and responsibilities seriously. They may not be sleeping well at night, and I know if I was responsible for this I wouldn't be, but that's so that we can.

Alan Nevill tweeted: "Hi, Steve. Regarding SpinRite 6.1, I use a Raspberry Pi 4 as a NAS. Will 6.1 be able to run on the Pi?" No. I replied to Alan, SpinRite is lovingly written in native x86 assembly language. So it is, and has always been, intimately tied to the Intel x86 processors. So there's no way that version 6.1, at least, of SpinRite will run on anything other than a system with an Intel processor. The only way to run SpinRite on a mass storage device from a non-Intel platform will be to temporarily move the device to any Intel PC. And of course the good news is that SpinRite 6.1 is so fast now that a great deal of testing and maintenance can be accomplished in very little time. But, you know, never say never. I don't know perfectly what the future of SpinRite is. But for the foreseeable future at least, even 7 and 7.x and everything that I know of in the future, it's still going to be Intel-based.

Someone whose handle is @fizch, he tweeted: "A few weeks ago, you mentioned a piece of networking equipment that you use for segmenting your network. For the life of me, I cannot find that episode of Security Now!, nor do I remember what the device was. Can you shed some light on this for me?"

And yes. I recall pointing up over my shoulder at the little SG-1100 firewall router by Netgate which runs pfSense and can do everything you might ever want a little router to do. Since it has separate physical LAN, WAN, and Auxiliary interfaces, it's perfect for creating an isolated IoT network whose devices can only see each other on the same sub-LAN, and they can see the Internet, but not across each other's network. And if you just google "netgate sg-1100," it'll take you right there. And literally it's what I'm using on my network right now. Yup, there it is. Neat little gizmo. And it's a turnkey pfSense firewall, and pfSense will do anything.

Ben Hutton tweeted: "@SGgrc," he says, "I'm a little behind, but in SN-856 you mention that you wouldn't want to be able to translate two-factor authentication generation data back into the original QR codes." He said: "But some apps allow for export and import. Could you explain how this is different, please?"

Okay. So a two-factor authenticator receives a QR code from a website to establish a two-factor identity for a user. The site will generate and store a random secret key which it associates with a user's account. It will then display that random key in the form of a QR code. The use of a QR code allows that key to be easily transmitted optically to an authenticator app on the user's smartphone using its camera. If we didn't have QR codes

or something similar, the site would need to have the user - it would have to display like the private key in hex or something, and have the user manually enter that key on the smartphone's touchscreen, which would be a nightmare. So using a QR code for this is a huge convenience. And at that moment, by the way, when the QR code is being displayed by the website, is when I press CTRL+P to capture and print that page for my own offline QR code authentication archive.

The reason I would rather use an authentication app that does not allow for private key export is that anyone who might be able to arrange to obtain brief unlocked access to my phone might export my collection of authenticator private keys, thus defeating all of the multifactor security I've come to rely upon. The only valid reason to ever need to export an authenticator's private key repository would be to move them to another authenticator app, for instance. But since I keep all of those keys printed on paper, I never need to export anything. I only need to re-import my private key archive, which is in the form of a sheaf of printed papers, each containing one QR code.

Bruce Schneier's famous advice regarding passwords applies here. Many years ago Bruce said: "Do not memorize your passwords. Write them down. They should be so complex that you cannot possibly remember them all, and we're very good at managing bits of paper," Bruce said. And note that printing these QR codes has the inherent advantage of moving the archive offline. It's true that someone with physical access who knew where this little sheaf of pages was stored could obtain all my authenticator QR codes. But that's not the threat model we're attempting to protect against. And if it were, safety deposit boxes or a safe or any sort of physical lock box is a long-established means for protecting that kind of real-world asset. The threat model we're attempting to thwart is online. So moving their backup offline provides for added protection.

And one last point is that I understand that it's not as high-tech to print QR codes on paper. But it's our technologies that often bite us in the butt. Sometimes, especially where security matters, a bit of judiciously placed low tech is the more secure solution, even if it's less glamorous.

Leo: And now, speaking of the possible or maybe the impossible, Global Privacy Control. Hit it, Steve.

Steve: Okay. So what is Global Privacy Control? Believe it or not, it is what was clearly the right solution and what I was vocally arguing for as being obviously the right solution when its predecessor, known as Do Not Track, was first proposed 13 years ago, back in 2009. Yes, during this podcast. Leo, we've been around for a while. We are finally going to get it. It will have legally enforceable teeth, and I could not be more pleased. I got a kick out of this. This is what Wikipedia has to say about Do Not Track, and it's useful because it also reminds us of a few things that happened.

Wikipedia says: "Do Not Track (DNT) is a no longer official HTTP header field, designed to allow Internet users to opt out of tracking by websites, which includes the collection of data regarding a user's activity across multiple distinct contexts and the retention, use, or sharing of data derived from that activity outside the context in which it occurred.

"The Do Not Track header was originally proposed in 2009 by researchers Christopher Soghoian, Sid Stamm, and Dan Kaminsky. Mozilla Firefox became the first browser to implement the feature, while Internet Explorer, Apple's Safari, Opera, and Google Chrome all later added support. Efforts to standardize Do Not Track by the W3C in the Tracking Preference Expression Working Group reached only the Candidate Recommendation stage and ended in September 2018 due to insufficient deployment and support.

"DNT," they write, "is not widely adopted by the industry, with companies citing the lack of legal mandates for its use, as well as unclear standards and guidelines for how websites are to interpret the header. Thus, critics purport that it is not guaranteed that enabling DNT will actually have any effect at all. The W3C disbanded its DNT working group in January 2019, citing insufficient support and adoption. Apple discontinued support for DNT the following month, citing browser fingerprinting concerns." Uh-huh. "As of November 2021, Mozilla Firefox continues to support DNT. In Firefox, the feature is turned on by default in private browsing mode and optional in regular mode.

"In 2020, a coalition of U.S.-based Internet companies announced the Global Privacy Control header that spiritually succeeds the Do Not Track header. The creators hope that this new header will meet the definition of 'user-enabled global privacy controls' defined by the California Consumer Privacy Act (CCPA) and European General Data Protection Regulation (GDPR). In this case, the new header would be automatically strengthened by existing laws, and companies would be required to honor it."

Okay. So that was DNT. What we will be getting is GPC. If I go to the globalprivacycontrol.org website, so it's globalprivacycontrol.org, website with Chrome, the top of the page looks like, and I have a picture of it in the show notes, there's a red dot, and it says "GPC signal not detected." And then, since this is the globalprivacycontrol.org site, it says "Please download a browser or extension that supports it." But if I go to the same site with Firefox, after enabling Firefox's already present and built-in global privacy control settings, I'm greeted with a different banner at the top. I get a green dot and the words "GPC signal detected." So from a technology standpoint GPC has 100% of the elegant simplicity that made DNT such an obviously correct solution.

The specification's abstract reads - this is the abstract for GPC. It reads: "This document defines a signal, transmitted over HTTP and through the DOM" - that's one additional feature that we'll talk about in a second - "that conveys a person's request to websites and services to not sell or share their personal information with third parties. This standard is intended to work with existing and upcoming legal frameworks that render such requests enforceable."

The specification's introduction explains. They say: "Building websites today often requires relying on services provided by businesses other than the one which a person chooses to interact with. This result is a natural consequence of the increasing complexity of web technology and of the division of labor between different services. While this architecture can be used in the service of better web experiences, it can also be abused to violate privacy." Which perfectly contextualizes this issue.

"Several legal frameworks exist, and more are on the way, within which people have the right to request that their privacy be protected, including requests that their data not be sold or shared beyond the business with which they intend to interact. Requiring that people manually express their rights for each and every site they visit is, however, impractical."

Okay. Then they cite a chunk of the California Consumer Protection Act, which reads - this is from the CCPA. "Given the ease and frequency by which personal information is collected and sold when a consumer visits a website, consumers should have a similarly easy ability to request to opt-out globally. This regulation" - that is, the California Consumer Protection Act now in force and in law in California, and the get out of free jail card is sunsetting and ending at the end of this year - says: "This regulation offers consumers a global choice to opt-out of the sale of personal information, as opposed to going website by website to make individual requests with each business each time they use a new browser or a new device." So in other words, thank goodness we're not going

to need to be doing something like clicking on "Yes, I'll accept your cookies," the equivalent of, wherever we go.

And then back to the specification for GPC, they said: "This specification addresses the issue by providing a way to signal, through an HTTP header or the DOM, a person's assertion of their applicable rights to prevent selling data about them to third parties or sharing data with them. This signal is equivalent, for example, to the 'global privacy control' in the CCPA regulations." And I'll get to that in a second.

So as I said, the specification could not be more clear or easy to implement. I have a sample header in the show notes. Whereas Do Not Track was DNT: 1 to enable it, this one is Sec, probably short for security, Sec- then GPC: 1. That's it. Your browser, when it makes a query to any asset, it's going to say GET as the verb, or maybe POST, and then the URL. Then there'll be a Host: header that tells the site which host it wants, so like example.com, and then Sec-GPC: 1 to say I am requesting of you the site associated with this query from my browser that you abide by these regulations. So it defines and standardizes this new HTTP header, Sec-GPC, which when affirmatively set has the value of "1," meaning "true."

Okay, now, for a minute, recall the mess that ensued when Microsoft decided to have Internet Explorer 10 default to transmitting that DNT: 1 header. Wikipedia reminds us of this. They wrote: "With IE10, when using the Express settings upon installation, a Do Not Track option was enabled by default for IE10 and Win8. Microsoft faced criticism for its decision to enable Do Not Track by default from advertising companies, who say that use of the Do Not Track header should be a choice made by the user and must not be automatically enabled. The companies also said that this decision would violate the Digital Advertising Alliance's agreement with the U.S. government to honor a Do Not Track system because the coalition said it would only honor such a system if it were not enabled by default by web browsers. A Microsoft spokesperson defended its decision, however, stating that users would prefer a web browser that automatically respected their privacy.

"On September 7, 2012, Roy Fielding, an author of the Do Not Track standard, committed a patch to the source code of the Apache HTTP Server, which would make the server explicitly ignore any use of the Do Not Track header by users of Internet Explorer 10. Fielding wrote that Microsoft's decision 'deliberately violates' the Do Not Track specification because it 'does not protect anyone's privacy unless the recipients believe it was set by a real human being, with a real preference for privacy over personalization.'

"The Do Not Track specification did not explicitly mandate that the use of Do Not Track actually be a choice until after the feature was implemented in Internet Explorer 10. According to Fielding, Microsoft knew its Do Not Track signals would be ignored, and that its goal was to effectively give the illusion of privacy while still catering to their own interests. On October 9th of 2012" - so he patched it on September 11th. On October 9th of 2012, just a little over one month later, Fielding's patch was commented out, restoring the previous correct behavior for Apache because nobody agreed with him.

"On April 3rd, 2015, Microsoft announced that starting with Windows 10, it would comply with the specification and no longer automatically enable Do Not Track as part of the operating system's Express default settings, but the company will 'provide customers with clear information on how to turn on this feature in the browser settings should they wish to."

Okay. So to prevent any of this, the formal specification for the new Sec-GPC header clearly states, they said: "A user agent MUST NOT generate a Sec-GPC header field if the person's Global Privacy Control preference is not enabled or defaulted to." But they allow it to be defaulted to. "A user agent MUST generate a Sec-GPC header field with a field

value that is exactly the numeric character '1' if the user's Global Privacy Control preference is set. A user agent MUST NOT generate more than one Sec-GPC in a given request and MUST NOT use a Sec-GPC field in an HTTP trailer." They're rare. Hardly anyone uses them. We talked about them once before.

"A server processing an HTTP request that contains a Sec-GPC header MUST ignore it and process the request as if the header had not been specified unless the field value is exactly the character '1.' If there are multiple Sec-GPC headers and at least one has a field value of '1,' then the server MUST treat the request as if there were only one Sec-GPC header with a field value of '1,' and as if there were none otherwise. HTTP intermediaries MUST NOT remove a Sec-GPC header set to '1,' but they MAY remove Sec-GPC headers that contain other values. Additionally, an HTTP intermediary that has reasons to believe that the person originating a given HTTP request has a do-not-sell-or-share preference MAY insert a Sec-GPC header set to '1.'"

So the specification also defines, as I noted, a new top-level JavaScript global value "globalPrivacyControl" which allows any script running in the browser to determine whether the query which loaded that script sent the Sec-GPC: 1 header to the server. And the sample script that they had, they have some JavaScript. They have an if statement with parens, and then they start with an exclamation point, bang, for negating. So if (!navigator.globalPrivacyControl), and then in curly braces they have the comment "wonderful, we can sell this person's data!" Meaning Global Privacy Control was not set in this browser or for this query.

Since the only thing that differentiates this from the earlier DNT effort is the promise of legal enforcement under the European Union's GDPR, and at least in California and soon Colorado and Connecticut, I was curious to look at that a bit more closely. Wikipedia closes out their discussion by noting: "GPC is a valid Do Not Sell My Personal Information signal according to the California Consumer Privacy Act, which stipulates that websites are legally required to respect a signal sent by users who want to opt-out of having their personal data sold. In July of 2021, the California Attorney General clarified through an FAQ that under law, the Global Privacy Control signal must be honored."

Now, as a California resident, this made me curious to read the actual California legislation. So I found it. But don't worry, I'm not going to drag everyone through it. I'll just note that I am completely satisfied with what the legalese says. I'm not an attorney, but it was written in very clear and concise English which, to my untrained eye, leaves zero wiggle room this time. They're not screwing around. For anyone who's curious to see for themselves, I have a link to the 28-page PDF where, on page 17 - and this is just an excerpt from the entire legislation, so this is a small chunk of it. That's why it's only 28 pages. But on its page 17, paragraph 999.315 is titled: "Requests to Opt-Out." And anyone who's interested can follow that.

So it appears that we're about to get what we wanted and hoped for more than a decade ago. If you're a Firefox user, bring up your browser's detailed configuration settings by entering "about:config" in the URL field, as we've talked about many times. Then enter "privacy.global" as the search term to reduce those gazillion items that will come up to just two. Then double-click on each to flip them from false to true, which will also make them bold. Then head over to https://globalprivacycontrol.org, and at the top of that page you'll see that your Firefox browser is now, and hopefully forever, transmitting your legally enforceable assertion that you do not give your permission...

Leo: It worked!

Steve: Yup, that you are affirmatively revoking permission and choosing to opt-out of any site that you visit's legal right to share your information with any other entity for any purpose whatsoever.

Leo: And they will honor it.

Steve: And, well, as individuals we don't need to enforce that right since once California's "right to cure" provision sunsets at the end of this year, it seems clear that the biggest violators of consumers' asserted legal rights to privacy will be taken to task by California's Attorney General, who has clearly stated their intention to do exactly that.

Now, although Chrome is becoming increasingly conspicuous for not directly supporting the transmission of this GPC signal, plenty of Chrome-compatible add-ons and other browsers already do: Abine, Brave, Disconnect, DuckDuckGo, OptMeowt and Privacy Badger.

Now, being a perennial minimalist, it would be nice if uBlock Origin were to add the GPC header for us since I already have uBlock Origin installed everywhere, including in Chrome. But Gorhill was asked about six months ago whether or not he would consider doing so, and he declined on the basis that it was just another DNT fiasco. That doesn't appear to be true, so there's some hope that in the future he might reconsider. Although he is extremely curmudgeonly, so perhaps not.

Leo: Highly unlikely.

Steve: Yeah. For me, I'm liking the new add-on "OptMeowt" where the "u" of "out" is instead a "w," making it "meow," as in cat's meow. So it's spelled O-P-T-M-E-O-W-T, OptMeowt.

Leo: OptMeowt.

Steve: Yeah. The reason is I like the way it looks, and it was developed by some guys who have been deep into the Global Privacy Control effort. Their names are listed as contributors to the spec. They're at Wesleyan University's privacy tech lab. Wesleyan's a private college in Middletown, Connecticut, probably not coincidentally where this legislation is on the verge of making them the third C, with California and Colorado preceding. OptMeowt is free and open source and hosted on GitHub. It's about 2.5MB, so it's a relatively small download. And their page at GitHub has a bunch of other privacy-related projects, for anyone who's interested.

It's available from the Chrome Web Store and supports all of the many Chromium-based browsers, so Chrome, Brave, Edge, Opera, and Vivaldi. The Brave browser began testing default inclusion of the GPC header last October, so it's probably well in place by now. And if you're a Brave user, just go to Global Privacy Control. I expect you're already going to see a green light shining back at you without needing to add anything else. So by all means, check to see whether your browser may already support it, like Firefox did, and I didn't know it and didn't have it turned on. Now I do, and it is.

So if your browser doesn't have it, this little OptMeowt looks like a slick way to add it. I've got a picture of its settings page here at the bottom of the show notes. It's just got three buttons: Enable, which sends Do Not Sell signals to every visited domain. Or you

can be selective if you like. You can give it a Domain List which sends Do Not Sell signals according to the custom Domain List which you can provide. Or you can Disable, does not send any Do Not Sell signals. So it took us a long time to get here, Leo, but this is certainly encouraging.

Leo: Yeah, yeah.

Steve: And, boy, this would just be great.

Leo: Yeah, because what we've had to do is develop tools that block tracking fingerprinting and things like that because companies just, you know, they're not going to...

Steve: Right. They want it.

Leo: They want it.

Steve: Yeah. If we don't tell them no, and if they don't suffer legal recourse in the event of ignoring what we're telling them, they're going to do it.

Leo: Yeah. That's awesome. That's awesome. So you think that thanks to Europe, Connecticut, and California, they're going to honor this because sites will have to.

Steve: Yes.

Leo: Great. Good.

Steve: Yes. Well, look what GDPR did to our cookies. And it's a pain in the butt.

Leo: I know. Everybody does it. Everybody does it.

Steve: Yes, exactly.

Leo: We're all scared to death of them.

Steve: Yup.

Leo: Even though it's a completely meaningless regulation.

Steve: And they will sue. I mean, and it looks like California is saying, look, you're going to honor this law, or you're not going to do business in California. Well, you can't not do business in California.

Leo: Right, right. Wow, that's great. Well, thank you, Steve. As always, some great information. And I am now going to go around to all my Firefoxes and do about:config and turn on those Global Privacy Controls. There's two settings. You just flip them to true. And you're right, it says I'm green light now, which is amazing. So another reason to like Firefox, although you said there are other browsers that do this. Chrome is the only one that won't, probably.

Steve: Yeah, I think Brave is - well, now apparently they did. They did support DNT at some point. So I think they'll end up saying, okay, fine, you know, we're not happy, but...

Leo: I hope so. I hope so, yeah.

Steve: Yeah.

Leo: Steve Gibson.

Steve: But Leo, that's big. I mean, that shuts everything down.

Leo: Yeah.

Steve: I mean, it's really big.

Leo: Yeah. It's kind of stunning, yeah. Wow. Globalprivacycontrol.org is the place to go to learn more about it.

Steve: Yup, yup.

Leo: Steve Gibson, see, now you know why you have to be here Tuesdays. You really do. Listen to the show. You don't have to listen to it live because you can get it of course downloaded from Steve's site. He's got a couple of unique versions. He has a 64Kb audio, that's the mainstream version. But he also has a 16Kb audio file for the bandwidth-impaired, and transcripts, which make it very easy to search for something you're interested in. You can even, correct me if I'm wrong, Steve, but I think you can search all the transcripts all at once with Google; right? So you can find...

Steve: You can use Google. And I also have a search box on GRC.

Leo: Perfect.

Steve: And it knows, in order to constrain it just to Security Now!.

Leo: So it'll get all of them, and you can find them. "Did Steve ever talk about" - yes, he did, and it's right here. That's great. Great stuff. GRC.com. While you're there, pick up SpinRite. We are inching closer, maybe even striding closer to version 6.1. 6.0 is current, though. And if you buy it at GRC.com you will be freely upgraded to 6.1 as long as you buy it now. You'll also get to participate in the upgrade to 6.1, so that's kind of fun. There's a forum and all sorts of stuff. GRC.com. Steve's also on Twitter, if you want to leave him a comment, @SGgrc, and his DMs are open. So slide on in, as the kids say.

We have on-demand versions of the 64Kb audio and video, strangely enough, on our website, which is TWiT.tv/sn. There's a YouTube channel full of videos. And you could subscribe in your favorite podcast client. That's another way to do it. Steve has a link to the RSS, as do we. Just paste that in, and then you'll get it the minute it's available.

If you do, if you are in a hurry, and you do want to watch it live, the freshest version of Security Now!, we do it on Tuesdays around 1:30 Pacific, 4:30 Eastern, 20:30 UTC at live.twit.tv. There's live audio and video. So you can listen at dinner or lunch, you know, and learn. Listen and learn, we call it. And if you are listening or watching live, please chat with us live at irc.twit.tv. I think I've now plugged everything.

Well, one more thing. There's I am told a few cabins left if you want to go with me and Paul Thurrott to Alaska in July on the Holland America Eurodam. It's going to be a great cruise. Lots of fun. We're going to see, I'm told we're going to see bears, otters, bald eagles, as well as glaciers, what's left of them: cruise.twit.tv. Still some cabins left. And I would like anybody who wants to go, to go. They have inexpensive cabins, if you want.

Steve: How long are you away from TWiT?

Leo: I'll just miss a week, so I'll miss one show, July 16th to the 23rd. We're going to go early, I think Thursday we're going to go up to Seattle, spend some time there. Mary Jo Foley tells about a good dive bar we can get wasted in, so that'll be fun. Then we'll get on the boat all hung over. I can't wait. It's going to be a lot of fun: cruise.twit.tv. You do not have to be a club member to go. By all means, everybody should join us. I invited you. But Steve said, "No, Lorrie won't let me go."

Steve: No.

Leo: So you're stuck on shore. We'll have to do a land tour for you, you and Lorrie. We'll figure something out. Thanks, Steve. Have a great week, and we'll see you next time.