

Security Now! #869 - 05-03-22

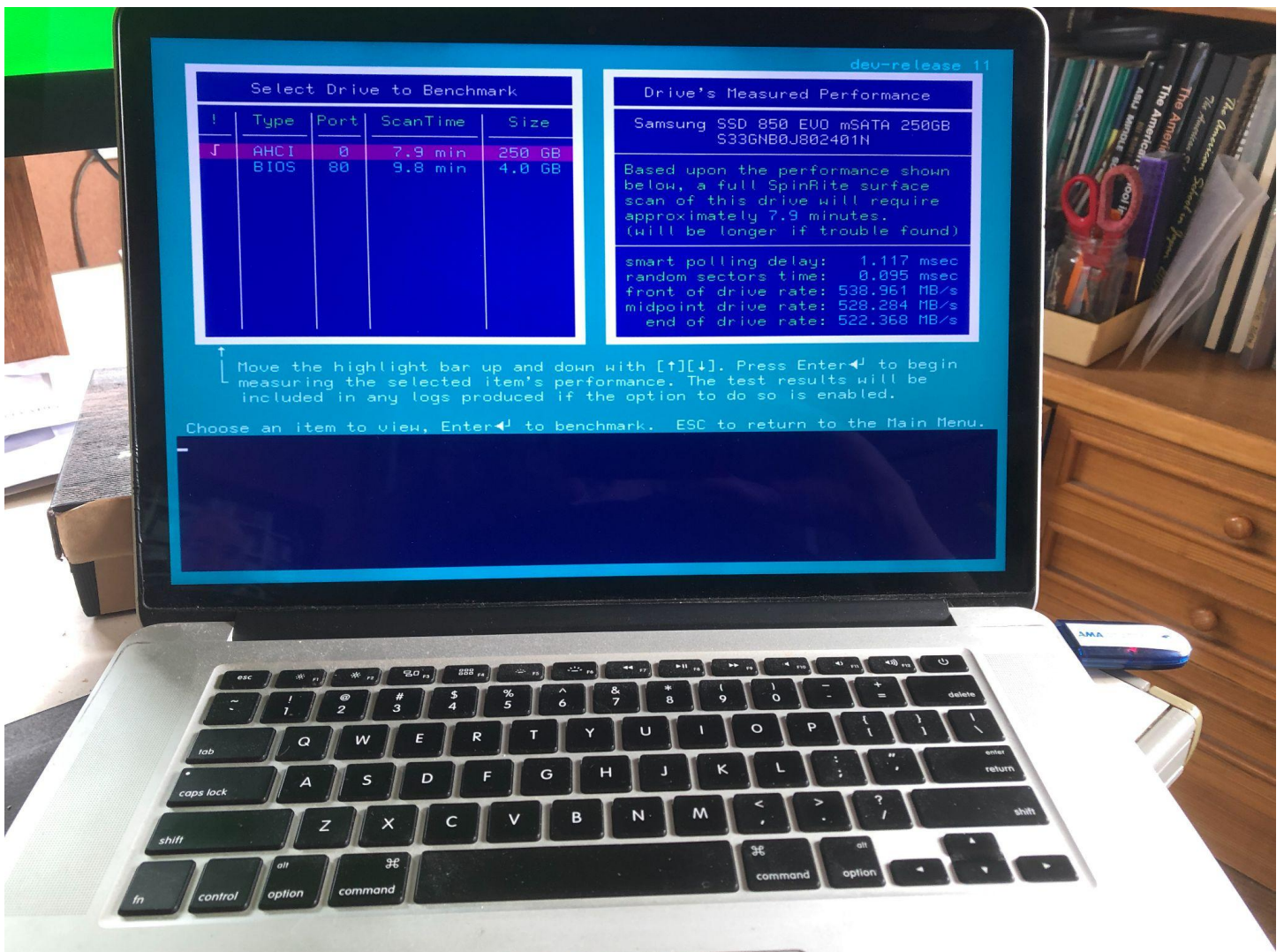
Global Privacy Control

(Hope Springs Eternal)

This week on Security Now!

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.

SpinRite v6.1 development pre-release #11 running on a MacBook...



Security News

DoD DIB-VDP Pilot Overview

<https://www.dc3.mil/Organizations/Vulnerability-Disclosure/DIB-VDP-Pilot/>

The program of the DoD was the "DIB-VDP Pilot Program." Despite suffering from program name abbreviation overload, that program just successfully finished its year-long pilot test.

The Defense Industrial Base Vulnerability Disclosure Program (DIB-VDP) Pilot is a 12-month voluntary event established collaboratively by DC3's DoD Defense Industrial Base Collaborative Information Sharing Environment (DCISE), DoD Vulnerability Disclosure Program (DoD VDP), and the Defense Counterintelligence and Security Agency (DCSA).

That year-long bug bounty program scrutinized a fraction of the U.S.'s massive defense industrial base. On Monday the effort's organizers announced that more than 400 valid vulnerabilities had been uncovered.

During the 12-month pilot, nearly three hundred vulnerability security researchers from HackerOne participated, filing 1,015 reports while examining the networks of individual participating defense contractors. As a result of their scrutiny, 401 vulnerabilities were deemed actionable and required remediation.

The pilot effort was run in coordination with 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 and over the course of the year it was expanded to include 41 entities and 348 systems.

What should give the program managers pause is that 401 actionable vulnerabilities were found in the 348 systems of 41 entities, yet somewhere between 100,000 to 300,000 companies contract directly with the Pentagon and its components. This legitimizes the long running worry about the latent digital vulnerabilities of the firms that make up the department'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 refer to last week, occurred back in 2009 when suspected Chinese hackers broke into one of the contractors working on the F-35 Joint Strike Fighter, the most expensive weapons system platform in U.S. history, and stole its design data.

Melissa Vice, the interim director of the DoD Vulnerability Disclosure Program, 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 (Defense Industrial Base) company assets with potential risks for critical infrastructure and the U.S. supply chain."

Armed with this data, an analysis of the DIB Vulnerability Report Management Network will now take place in order to document the pilot program's lessons learned and inform the way forward for a funded program.

Alex Rice, co-founder and CTO of HackerOne 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 VDPs (Vulnerability Disclosure Program) in the real world."

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 the sprawling DoD enterprise. Earlier this year the DOD's CIO, John Sherman, said the Pentagon was "getting rolling on both of those" examinations. Now we've seen, and the bureaucracy has witnessed, the clear feasibility of these programs.

When you stop to think about it, this approach makes so much sense. No sane and talented white hat hacker one is going to poke at government networks and systems at the risk of triggering a law enforcement action against them. We've seen exactly that happen in the past. So that must be done within the bounds of a sanctioned bug hunting program. And 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 enterprises' networks and systems. So the government needs to pay. And no private enterprise has as much money lying around as 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 the private sector become a permanent reality in the future. Successes such as were seen during this "DoD DIB-VDP Pilot" — though they really do need a sexier name for it — move us another step in that direction.

The OpenSSF and the Package Analysis project:

We've never talked about the relatively recently-formed OpenSSF. OpenSSF stands for Open Source Security Foundation, it's a Linux Foundation project created in 2020 by a large group of significant tech and financial firms with the charter to help steer, guide, and share open-source security tools.

Its membership is divided into Premiere and General members with the premiere 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, RedHat, Snyk, SonaType, VMWare and WiPro. There are about three times as many General members, which I won't enumerate, but many are notable such as Bloomberg, Comcast, Goldman Sachs, MongoDB, Spotify, Suse, Tenable, Tencent, Carnegie Mellon & Mitre.

The website homepage of this relatively new Foundation (<https://openssf.org/>) states: "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 they divide this effort is divided into three components:

- **Working Groups:** Collaborate on the planning, design, and delivery of security tooling and best practices that secure critical open source projects.

- **Town Halls:** Stay informed about the latest happenings in open source security and engage with experts in our community,
- **Training:** Take free courses on secure coding practices as part of our Software Development Fundamentals Professional Certificate.

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, of their project known as the **Package Analysis Project:**

<https://openssf.org/blog/2022/04/28/introducing-package-analysis-scanning-open-source-packages-for-malicious-behavior/>

*Introducing **Package Analysis: Scanning open source packages for malicious behavior***

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. In just one month of analysis, the project identified more than 200 malicious packages uploaded to PyPI and npm.

Okay, now 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 of what they found. Their announcement continues:

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're 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 rules of thumb, tweaking and adjusting of weighting factors is necessary. But if the signals they are able to obtain from their static code analysis are strong enough, robust determinations should be possible. They continue:

The vast majority of the malicious packages we detected are dependency confusion and typosquatting attacks. The packages we found usually contain a simple script that runs during an install and calls home with a few details about the host. These packages 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.

There are lots of opportunities for involvement with this project, and we welcome anyone interested in contributing to the future goals of:

- *detecting differences in package behavior over time;*
- *automating the processing of the Package Analysis results;*
- *storing the packages themselves as they are processed for long-term analysis;*
- *and improving the reliability of the pipeline.*

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 and Sigstore, which expand beyond the security of packages themselves to address package integrity across the supply chain.

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 takedown.

I mentioned some specific examples from among the more than 200 malicious packages discovered during the first month of the new system's use:

https://github.com/ossf/package-analysis/blob/main/docs/case_studies.md

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. Stolen accounts can be resold, used for fraud (e.g. cryptocurrency scams), or used to gain insider knowledge or secrets.

PyPI: discordcmd / 2022-02-18, Analysis Result:

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. 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:

NPM: colorsss / 2022-03-05, Analysis Result:

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:

- *searches through local browser databases for a token;*
- *queries the Discord server to discover details about the token;*
- *and exfiltrates these - details to a Discord server controlled by the attacker.*

They also uncovered a couple of remote shells. They explain that: "A remote shell is used by an attacker to provide access to a command shell running on a target machine over the network. These are usually "reverse shells" that connect back to an attacker-controlled machine."

NPM: @roku-web-core/ajax / 2022-03-08, Analysis Result

During install, this NPM package exfiltrates details of the machine it is 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.

PyPI: secrevthree / 2022-03-14, Analysis Result

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.

NPM: random-vouchercode-generator / 2022-03-14, Analysis Result

When the library is imported it queries a server running on the 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, 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. This package was discovered from the unusual request to the Heroku server.

Their case studies page finishes by talking about Dependency Confusion & Typosquatting. They wrote:

The vast majority of the malicious packages we detect are dependency confusion and typosquatting attacks.

The packages found usually contain a simple script that runs during install and calls home with a few details about the host. These packages 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.

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 back attacks. The same domains appear across unrelated packages and have no apparent connection to the packages themselves. Many packages also used unusual version numbers that were high (e.g. v5.0.0, v99.10.9) for a package with no previous versions.

As we've covered previously, the use of high version numbers with dependency confusion attacks is to trick package management software into believing that a new and better release of a package is available, which it will then substitute in favor of a lower-version but authentic package.

Google, an active participant in this effort added a bit of additional background in their own blog posting. They explained:

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, package repositories have limited resources to review the thousands of daily updates and must maintain an open model where anyone can freely contribute. As a result, malicious packages like ua-parser-js, and node-ipc are regularly uploaded to popular repositories despite their best efforts, with sometimes devastating consequences for users.

Google, a member of the Open Source Security Foundation (OpenSSF), 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 on. 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.

And Google's blog posting concluded:

The short time frame and low sophistication needed for finding the results above underscore the challenge facing open source package repositories. While many of the results above were likely the work of security researchers, any one of these packages could have done far more to hurt the unfortunate victims who installed them.

These results show the clear need for more investment in vetting packages being published in order to keep users safe. This is a growing space, and having an open standard for reporting would help centralize analysis results and offer consumers a trusted place to assess the packages they're considering using. Creating an open standard should also foster healthy competition, promote integration, and raise the overall security of open source packages.

Over time we hope that the Package Analysis program will offer comprehensive knowledge about the behavior and capabilities of packages across open source software, and help guide the future efforts needed to make the ecosystem more secure for everyone. To get involved, please check out the GitHub Project and Milestones for opportunities to contribute.

This is all for the good. The bad news is that this again puts us back into a 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 example from those descriptions was: *"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 back attacks."* Once the bad guys learn that these simple factors, which did not previously interfere with their nefarious aims have become problematic, they'll simply stop being 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 before. Now it is, so it'll happen.

We already saw something similar with the adoption of package submission bots being used to create unique per-package-submission accounts. This prevented casual linkage to other malware which had also been submitted by the same miscreant. The underlying problem is that, at the moment, anyone is able to offer packages for submission. There is no reputation system in place. And the problem with reputation-based filtering is that it can be discriminatory, preventing valid unknown publishers, who have 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 the "OpenSSF" — the Open Source Security Foundation to work towards introducing improvements in this valuable ecosystem.

Connecticut moves toward state privacy protections.

The U.S. Federal Government is not moving forward on consumer privacy protections, so several states are.

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 state law by Connecticut's governor, Ned Lamont, Connecticut would join four other states which have already enacted similar legislation. Those four others are: California, Colorado, Virginia and Utah.

Connecticut's bill, 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, it would take effect July 1 of NEXT year (2023). And it closely resembles the privacy laws 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.

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, so that consumers can opt out of data sales at all companies in a single step. And as we know, "Global Privacy Control" is today's main topic. So we'll be dissecting it in detail shortly.

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, a federal privacy law remains necessary to ensure that all Americans are guaranteed strong, baseline protections for the processing of their personal information."

The law also joins California and Colorado in adding sunset clauses to the "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. The "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. And its presence has been a hotly debated topic in many state assemblies across the country. It has 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 a "get out of jail free" card for companies violating consumer privacy. However, Connecticut's right to cure provision sunsets on December 31, 2024. Colorado's provision sunsets the next day on January 1, 2025 while California's sunsets the first of NEXT year, on January 1, 2023. Once these sunset, the states will be able to take enforcement action against organizations that violate the law.

California, Colorado and Connecticut are being referred to as the 3Cs and a lawyer, David Stauss, who serves as chair of the HuschBlackwell 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 US 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.

Connecticut's SB6 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 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.

The 3Cs won't care and won't need to. Any web-based business with a customer in California, Colorado or, presumably Connecticut will need to get their digital act together soon — in the case

of California, 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 "I want privacy!" at every website we visit.

Closing The Loop

dstacer / @dstacer

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?

That is a darn good question. I had to think a bit about how to explain why I still feel that both of those apparently opposing evaluations can 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. 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 lookup every site we were visiting, then display, copy and paste our username and password for that site into the browser. 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 #1 attack targets. They are the "attack surface" which we all present to the Internet as we move about. They are not perfectly secure. They are not as secure as we could wish they were. Yet we are not seeing reports of browser native password databases, nor 3rd-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 3rd-party password managers, are really taking their jobs and responsibilities seriously. They may not be sleeping well at night so that we can.

Alan Nevill / @AlanNevill

Hi Steve, Re. Spinrite 6.1. I use a Raspberry pi 4 as a NAS. Will 6.1 be able to run on the pi?

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 of SpinRite will run on anything other than a system with an Intel. The only way to run SpinRite on the mass storage devices of non-Intel platforms will be to temporarily move the device to any Intel PC. The good news is that SpinRite v6.1 is so fast that a great deal of testing and maintenance can be accomplished in very little time.

fizch / @fizch

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?

Yes. I recall pointing up over my shoulder at the little SG-1100 firewall router by NetGate which runs pfSense and can do anything you might ever want a little router to do. Since it has separate WAN, LAN and AUXiliary interfaces, it's perfect for creating an isolated IoT network whose devices can only see each other and the Internet, but not into the more secure LAN network.

<https://shop.netgate.com/products/1100-pfsense>

Ben Hutton / @relequestual

@SGgrc I'm a little behind, but in SN 856, you mention that you wouldn't want to be able to translate 2FA generation data back into the QR codes. But, some apps allow for export and import. Could you explain how this is different please?

A two-factor authenticator receives a QR code from a web site to establish a two-factor identity for a user. The site will generate and store a random secret key for the 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 authentication 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 enter a private key manually on the smartphone's touch screen, which would be a nightmare. So using a QR code for this is a huge convenience. And THAT, by the way, is when I press Control-P to capture and print that page for my own offline authentication archive.

The reason I would rather use an authentication app that does not allow for private key export is that anyone who could arrange to obtain brief unlocked access to my phone might export my collection of authenticator private keys, thus defeating all of the multi-factor security that I have 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 or 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 pages, 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 about managing bits of paper.”

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 of 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 real-world assets. 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, even if less glamorous, solution.

Moxie on Port Knocking

Moxie Marlinspike, is most famous for his hand in the development of Signal, which, as we know, provides a secure technology for true end-to-end encryption of instant messages. And unlike Telegram, Signal employs standard proven and well understood cryptographic primitives.

It turns out that 10 years ago Moxie took a stab at his own design and implementation of port knocking. For anyone who’s interested, it’s on GitHub: <https://github.com/moxie0/knockknock> I won’t go into it in depth. But what I loved, and the reason I’m mentioning it, 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.

Again, you are running network services with security vulnerabilities in them.

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?

*The name of the game is to isolate, compartmentalize, and **expose running services as little as possible**. That's where knockknock comes in. Given that your network services are insecure, you want to expose as few of them to the world as possible. I offer knockknock as a possible solution to minimizing exposure.*

Global Privacy Control

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.

We are finally going to get it. It will have legally enforceable teeth, and I could not be more pleased. Here’s what Wikipedia has to say about “Do Not Track”:

Do Not Track (DNT) is a no longer [an] 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 (so-goy'-an), 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 (DNT) Working Group reached only the Candidate Recommendation stage and ended in September 2018 due to insufficient deployment and support.

DNT 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. 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 US-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.

That was DNT. What we will be getting is GPC. If I go to the <https://globalprivacycontrol.org/> Website with Chrome, the top of the page looks like this:

● GPC signal not detected.

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 this:

● GPC signal detected.

Test against the reference server.

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 document defines a signal, transmitted over HTTP and through the DOM, 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:

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.

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.

[The introduction then cites a paragraph from the California Consumer Protection Act:]

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 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. [CCPA-AG-FINAL-STATEMENT]

[In other words, thank god we're not going to need to be doing something like clicking on "Yes, I'll accept your cookies" wherever we go.]

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 [CCPA-REGULATIONS].

And the specification could not be clearer or more easy to implement:

GET /something/here HTTP/2

Host: example.com

Sec-GPC: 1

It defines and standardizes a new HTTP header "Sec-GPC" which, when affirmatively set, has the value of "1" meaning "true". Now, recall the mess that ensued when Microsoft decided to have Internet Explorer 10 default to transmitting the DNT:1 header. Wikipedia reminds us, writing:

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 an illusion of privacy while still catering to their own interests. On October 9, 2012, Fielding's patch was commented out, restoring the previous behavior.

On April 3, 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 that the company will "provide customers with clear information on how to turn this feature on in the browser settings should they wish to do so".

So, to prevent any of this, the formal specification for the Sec-GPC header clearly states:

- 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.
- A user agent MUST generate a Sec-GPC header field with a field-value that is exactly the numeric character "1" if the person's Global Privacy Control preference is set.
- A user agent MUST NOT generate more than one Sec-GPC in a given HTTP request and MUST NOT use a Sec-GPC field in an HTTP trailer
- A server processing an HTTP request that contains a Sec-GPC header MUST ignore it and process the request as if that 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 exactly "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".

The specification also defines a new top-level JavaScript global value "globalPrivacyControl" which allows any scripting running in the browser to determine whether the query which loaded that script sent the Sec-GPC:1 header to the server:

```
if (!navigator.globalPrivacyControl) {  
    // wonderful, we can sell this person's data!  
}
```

Since the only thing that differentiates this from the earlier DNT:1 effort is the promise of legal enforcement under the European Union's GDPR and at least in California and soon in 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 (CCPA), 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 2021, the California Attorney General clarified through an FAQ that under law, the Global Privacy Control signal must be honored.

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.

For anyone who's curious to see for themselves, I have a link to the a 28-page PDF where, on page 17 paragraph § 999.315 is titled: Requests to Opt-Out:

<https://www.oag.ca.gov/sites/all/files/agweb/pdfs/privacy/oal-sub-final-text-of-regs.pdf>

Okay. So it appears that we're about to get what we wanted more than a decade ago. If you are a Firefox user, bring up your browser's detailed configuration settings by entering "about:config" in the URL field. Then enter "privacy.global" as the search term to reduce the gazillion items to just two. 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, that you are affirmatively revoking permission and choosing to opt-out of any site you visit's legal right to share your information with any other entity for any purpose whatsoever. And as individuals we won'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 consumer's 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.

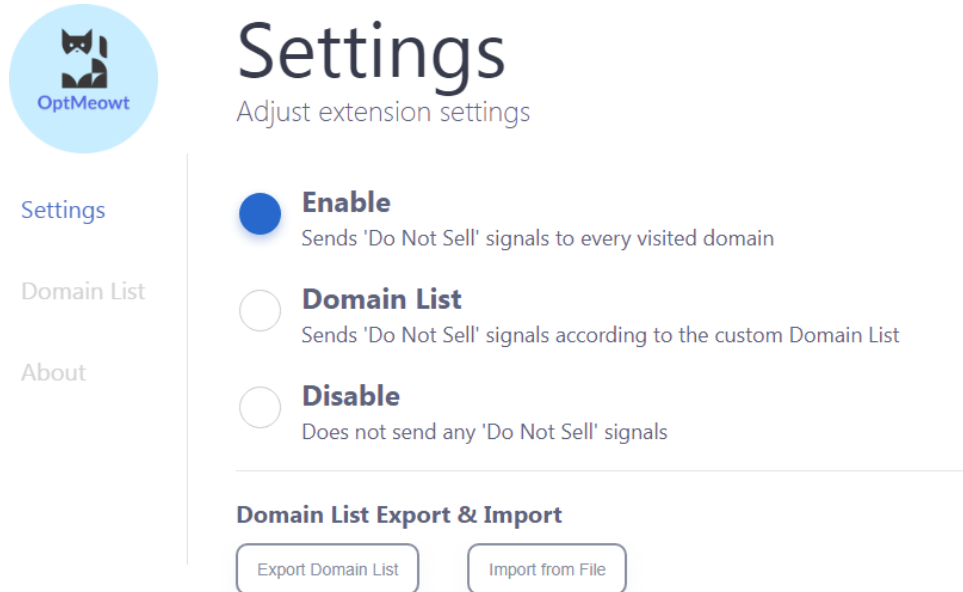
Although Chrome is becoming increasingly conspicuous in not directly supporting the transmission of this GPC signal, plenty of Chrome-compatible add-ons and other browsers already do: Abine, Brave, Disconnect, DuckDuckGo, OptMeowt and Privacy Badger.

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 six months ago whether or not he would consider doing so and he declined on the basis that it was just another DNT:1 fiasco. That doesn't appear to be true, so there's some hope that in the future he might reconsider... though he is extremely curmudgeonly, so perhaps not.

For Chrome, I'm liking the add-on "OptMeowt" where the 'u' of 'out' is instead a 'w' making it "meow" as in a cat's meow. So it's spelled O P T M E O W T. The reason I like the way it looks is that 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 is a private college in Middletown, Connecticut. OptMeowt is free and open source and hosted on GitHub. And their page at GitHub has a bunch of other privacy-related projects:

<https://github.com/privacy-tech-lab/>

Optmeowt is available from the Chrome Web Store and supports all of the many Chromium-based browsers. So, Chrome, Brave, Edge, Opera, Vivaldi. The Brave browser began testing the default inclusion of the GPC header last October, so it's probably well in place by now. And other privacy focused browsers may have also done the same. So you might check your Chromium-based browser for its support first, which you can do by just going over to <https://globalprivacycontrol.org/> and seeing what the top of the page says.



Settings
Adjust extension settings

Enable
Sends 'Do Not Sell' signals to every visited domain

Domain List
Sends 'Do Not Sell' signals according to the custom Domain List

Disable
Does not send any 'Do Not Sell' signals

Domain List Export & Import

Export Domain List Import from File

It took us a long time to get here, but this is certainly encouraging.

