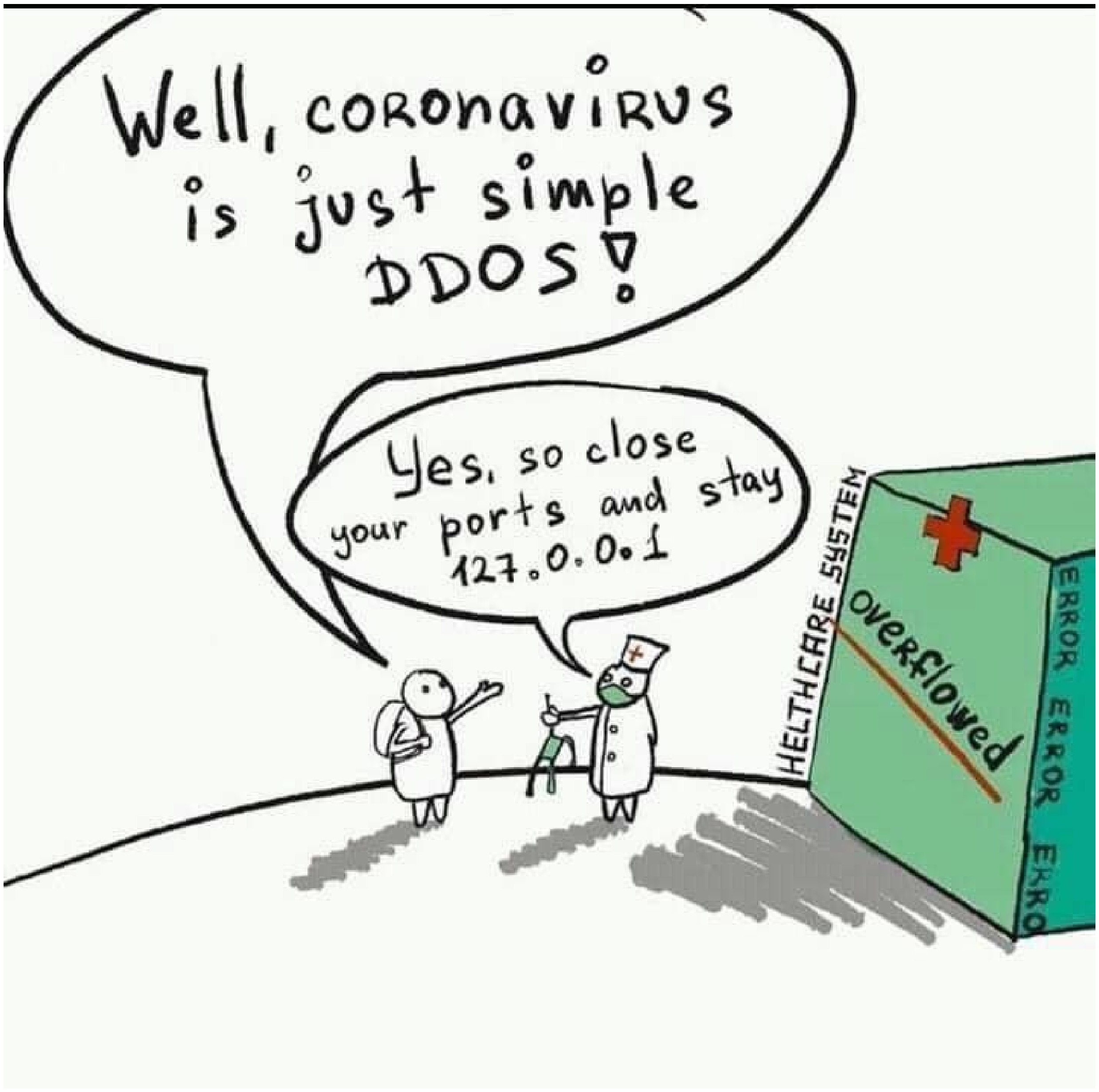
Security Now! #760 - 03-31-20

Folding Proteins

This week on Security Now!

This week we examine some consequences of increased telecommuting with the use of RDP and VPNs skyrocketing, we examine a new bug in iOS's handling of VPN connections. We also look at Google's unrelenting quest to get the "www" out, some changes to Firefox and further revisions of browser release schedules. We take a deep dive into a very welcome forthcoming code security feature for Windows 10, we share an action-item for users of OpenWRT routers and the result of an audit of Cloudflare's privacy-enforcing DNS service. Then we update our listeners on a few interesting bits of feedback, SQRL and SpinRite miscellany and finish by examining a new opportunity to donate our unused CPU cycles for help with COVID-19 research.


Well, coronavirus is just simple DDOS!
Yes, so close your ports and stay 127.0.0.1
HELTHCARE SYSTEM
overflowed
ERROR ERROR ERRO

Security News

RDP and VPN use skyrocketing

To no one's surprise, the use of the cloud and all remote access technologies has jumped up significantly since the "stay at home" orders went into effect at many parts of the country.

To put some numbers behind this, the appearance of RDP servers, which is, of course, Microsoft's perpetually security-challenged remote desktop protocol has jumped 41%. And popular VPN port appearances have jumped 33%.

According to data compiled by Shodan, one of several often quoted public Internet-wide port search engines, the number of publicly visible RDP endpoints has gone from roughly 3 million at the start of the year to nearly 4.4 million the day before yesterday, Sunday, March 29, 2020.

Also, this data only reflects RDP servers listening on the standard RDP port 3389. John Matherly, the Shoran's CEO and founder, noted in an interview with ZDNet, that a similar surge has also been seen on port 3388 (3389 minus 1) which is regularly used by system admins to hide the RDP service from attacks. For that not well hidden port, the number has also jumped, in this case by 36.8%, from roughly 60,000 at the start of the year, to over 80,000 now.

And I know I'm a broken record about this, but seeing a jump in exposed RDP ports is horrifying, since ALL of our experience informs us that Microsoft has never managed to make RDP safe to expose publicly. RDP should always be tucked safely behind a strong VPN, which itself uses some form of multifactor authenticator... as all good strong enterprise VPNs do.

And speaking of VPNs, Shoran has seen the number of servers running VPN protocols like IKE and PPTP jump up by a third, from 7.5 million systems, to nearly 10 million now. IKE and PPTP are typically enterprise VPNs which serve as gateways to intranet and corporate networks.

The use of consumer-grade VPNs has also seen a sudden increase usage, though in the case of the consumer it's likely for use in bypassing geo-fencing access to online content while people are stuck at home. ZDNet reported that last week NordVPN saw a 165% rise in users since March 11, while Atlas VPN reported a 124% surge in VPN use among its US userbase. As we know, global VPN providers can allow their users to have a "POP" -- a point of presence -- in other locations. Though there's something of a cat-and-mouse game with this, since the providers are aware of this, too. For example, the VPN rating site "Top10VPN" separately rates VPN as "Netflix VPNs" and has this to say:

> *To bypass Netflix's location restrictions and unblock all the 'hidden' TV series and movies, you need to know what's the best VPN for Netflix. Unfortunately, not all VPNs work with Netflix. But don't worry — we're here to help.*
>
> *Our team of experts regularly tests 72 VPNs to see if they unblock Netflix libraries in the US, the UK, and in many more countries.*
>
> *The VPNs we recommend below will give you reliable buffer-free access to Netflix from anywhere in the world — granting you free access to the Studio Ghibli movies, Friends, The Office, and so much more.*

Top10VPN also notes that the US has seen a recent 65% overall increase in consumer-use of VPN. https://www.top10vpn.com/news/vpn/covid-19-vpn-demand-statistics/

VPN handling bug in iOS v13.4 (and also iOS v13.3.1)
And speaking of VPNs, the most recent release of iOS v13.4 has been in the security community news due to a mistake in iOS's VPN handling that was first discovered by the ProtonVPN folks who reported it to Apple and simultaneously to their own users. Typically, Proton would wait 90 days before exposing a flaw in third-party software through its responsible disclosure program. However, they felt that they needed to alert their own VPN users to the vulnerability.

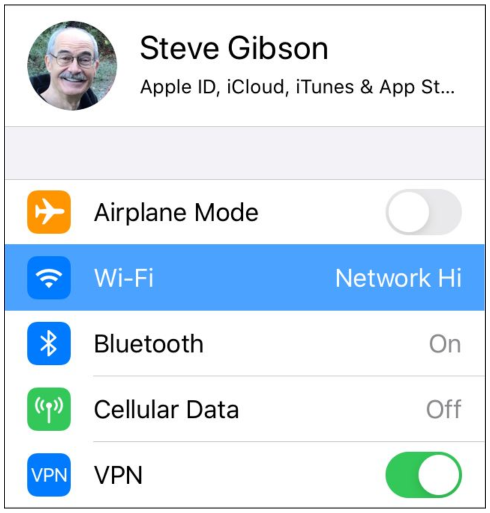
Okay, so what's the problem?

A ProtonVPN researcher was using Wireshark to monitor the packet traffic from an iOS device when he noticed that even after the VPN was turned on and its tunnel was brought up, non-tunneled traffic continued to be exchanged with the iOS device. iOS was not closing its existing connections and then reconnecting them once the VPN was in place as it should have been.

So, things that were started after the VPN was active will be securely routed through the VPN. But pre-existing connections would not be protected unless the individual services close or reset the connection themselves. From a practical standpoint, as we know, nearly everything these days is HTTPS, so data would not be in the clear. But that's often not the point, since the user's true IP address would continue to be exposed... which is not what anyone using a VPN expects and probably wants.

The ProtonVPN folks found a simple brute force workaround, though it's a bit annoying: After bringing up the VPN, kill all connections by briefly enabling and then disabling "Airplane" or "Flight" mode.

Apple's suggestion is to configure the Always-on VPN setting via mobile device management (MDM). But it takes some work with putting the device into supervised mode and using the Apple Configurator. I have one of my iPads that is VPNed like that. I think that I must have done it years ago when we were talking about it on the podcast, but I no longer recall the details.


Steve Gibson
Apple ID, iCloud, iTunes & App St...

Airplane Mode	Off
Wi-Fi	Network Hi
Bluetooth	On
Cellular Data	Off
VPN	On

I'm sure Apple will fix this soon. In the meantime, when you bring up a VPN tunnel, once the tunnel is up and established, switch off all WiFi communications using “Airplane” mode, then turn it back on.

To 'www' or not to 'www'

Talk about a ridiculous and fraught issue: Google has continued to battle the industry and its own users over the display of the 'www' in URLs, which it insists upon calling a "trivial" subdomain.

It was back in 2018 that Google first announced its intention to have Chrome 69 "elide", as they put it, the "www" from Chrome's displayed URL. The uproar of opposition caused them to rethink that decision. But someone, somewhere, is quite stubborn about this and they didn't let it go. Last August, with the release of Chrome 76, the 'www' was finally disappeared from Chrome's URL display. And many vocal people are very unhappy.

At the time, Google wrote: "The Chrome team values the simplicity, usability, and security of UI surfaces. To make URLs easier to read and understand, and to remove distractions from the registrable domain, we will hide URL components that are irrelevant to most Chrome users."

Conversely, typical complaints are things like this: "It causes confusion in that what the user sees as the URL in the omnibox is not reflected in the actual value when copied, it does not match the SSL certificate, and there are many sites that do not automatically map the naked domain to www."   In other words, the "www" is often not superfluous.

Which brings us to today. There must have been a large enough outcry over the loss of the “www” and the “https” to again cause Google to rethink it. So, Chromium developers are now testing a new Omnibox context menu that would give users the option to “Always Show Full URLs.”

This appears to still be a bit begrudging. Google continues to believe that showing what it calls a “trivial subdomain” will distract users when making security assessments. The feature is in the Chrome 83 Canary build, appearing in a context menu of the so-called "Omnibox" where the URL is displayed. After it has been set, that setting will be retained to always display the full web address including the “https” and “www”.

The Chromium developers outlined their plan for users to opt-out of URL snippage in a post on the bug tracker titled “Implement Omnibox context menu option to always show full URLs”. That post’s author, a Chromium software engineer Livvie Lin, wrote:

> The Omnibox context menu should provide an option that will prevent URL elisions for the entire Chrome profile. However, showing the full URL may detract from the parts of the URL that are more important to making a security decision on a webpage.

So, Chrome will continue to suppress the display of “https” and “www” by default. But it will also finally address its detractors by offering a context menu with a sticky setting to show the entire URL.

Firefox 76 to finally stop assuming "http"
Our listeners will know that this is something I've been commenting about being wrong for some time. With the majority of websites now HTTPS, and a definite bias in favor of moving everything to HTTPS, it has seemed wrong to me that if someone were to only enter "amazon.com" into their web browser, their browser would assume the prefix of "http://." Sure, once upon a time. But today?

On the Bugzilla page, where bugs go to die and new features are born as bugs because they don't yet exist, there's an entry for "Bug 1613063" flagged as experimental and titled: "HTTPS Only Mode". The description reads:

> Currently, if a Firefox user types foo.com in the address bar then our internal machinery establishes an HTTP connection to foo.com. Within this project we will expose a preference which allows end users to opt into an 'HTTPS Only' mode which tries to establish an HTTPS connection rather than an HTTP connection for foo.com. Further, we will upgrade all subresources within the page to load using https instead of http.
>
> Implementation considerations:
> - For top-level loads which encounter a time-out we could provide some kind of error page with a button which would allow the end user to load the requested page using http.
> - For subsource loads we could fail silently and just log some info to the console.

The setting will initially be off by default and I'm sure Mozilla's instrumentation will monitor the success of those who turn it on. Then, at some point, assuming that all goes well the default may be flipped to "On" and "https://" will become the assumed protocol when neither are supplied.

Google again revises its schedule for Chrome releases.
As we noted last week, Google announced that it would be canceling the release of Chrome 81 due to the disruptions created by COVID-19 and would only be focusing upon security fixes for Chrome 80. Apparently now that's changed...

In the Chrome blog last Thursday, Google said it plans to resume work on Chrome releases.

Google said that I keeping with its plans, the current Chrome 80 release will be receiving security updates But also that Chrome 81, which was originally scheduled for release on March 17, has now been rescheduled for release on April 7th, at which time, web developers and system administrators would have had the time to adapt to their new working conditions.

This does result in dropping release 82 altogether with its new features into release 83 and future releases. And release 83 is now expected to be released sometime around mid-May.

Last week we noted that Edge was also pausing in sync with Chrome. And last week, citing the impact on its customer base from COVID-19, Microsoft announced that starting in May, optional Windows 10 cumulative updates would also be paused.

And as we know, while Mozilla didn't pause Firefox updates, it did roll back the termination of TLS 1.0 and 1.1, which would have prevented their users from accessing HTTPS-based government websites that might have contained COVID-19-related information.

Microsoft moves to support “Shadow Stacks”

Last Wednesday, Microsoft's Hari Pulapaka, their Group Program Manager for the Windows Kernel, updated the world on the state of Microsoft's plans to add Hardware-enforced Stack Protection to Windows. In this case Hardware-enforced Stack Protection takes the form of Shadow Stacks, which we've talked about in the past. It requires hardware support from Intel in the form of Intel's Control-flow Enforcement Technology (CET).

A stack is one of the key innovations in the design of CPU architectures and stacks have been around for a long time, though not forever. For example, the earliest DEC PDP minicomputers did not have any stack. When I was writing those demo programs for the PDP-8, the lack of a stack was an annoyance. But DEC fixed that with the 36-bit PDP-10 and the 16-bit 11 which were both much more powerful minicomputers.

The main feature of a stack is that it's visible to the programmer. The processor has a pointer to the top of the stack, called the stack pointer, and there are instructions for pushing various amounts of data onto the stack and popping it off the stack. When parameters are passed to a procedure in a procedural programming language, the parameters are typically passed by having the caller push their values onto the stack before calling the procedure. Then, when actually calling the procedure, the CPU pushes the caller's return address onto the stack as it jumps to the beginning of the procedure. And if the procedure uses local variables, space will be allocated for them by moving the top-of-stack downward to create a buffer region to contain the local temporaries. The point is that the stack (a) serves as a sort of multi-function scratchpad which does a very efficient job of giving transient data a place to live and (b) the stack is visible to the programmer, whose caller parameters and local variables are all present. And, as we know too well, programmers all too often allocate temporary communications buffer space on the stack. But when all goes well, a stack is an incredibly elegant innovation for CPUs.

Okay, so what's a shadow stack? I deliberately noted the many different sorts of things that cohabitate and share the single stack. As we have seen, historically, the biggest danger is that the control-flow data -- subroutine return addresses -- share the stack with the many other sources and forms of non-control-flow data.

And remember that ALL OF IT is visible to the programmer. If a programmer wished to cause his subroutine to jump somewhere other than back to its caller, that's trivial to arrange. Simply overwrite the correct subroutine return address with any other address in the system and upon exiting from the subroutine the CPU will dutifully read that modified address from the stack and jump there. The point is, if malicious code somehow managed to arrange to do exactly that, we have opened up a huge vulnerability.

By comparison, a shadow stack lives in the shadows. Unlike the primary system stack it is NOT visible to the programmer. And this means that it's also not visible to any malicious code that might get loose. Intel has added this “shadow stack” feature to their future CET-equipped CPUs.

Unlike the main system stack, which contains a wonderfully dynamic hodge podge of data and control-flow, the shadow stack contains only return addresses. When a program makes a call to a subroutine procedure, the CET-equipped CPU pushes the return address onto BOTH the main system stack and the invisible shadow stack. The main stack receives and holds all manner of other information. But the shadow stack contains only return addresses. When the called procedure returns, the CET-equipped CPU pops the return address from the main stack and also separately from the shadow stack and compares them. They are guaranteed to match so long as nothing nefarious has modified the return address on the system stack. But they will not match if anything might have modified or overwritten the original value stored on the system stack. So this is a very slick way to enable Intel's CPU hardware to catch any of the very common stack overwrite mistakes as well as buffer overruns and other common stack-based security failures.

And... although support for this has been lagging from Intel, last Wednesday's announcement was that support for this technology is now under development and a preview is available in the Windows 10 Insider preview builds (fast ring).

Intel's specification for this has been public for several years and support for it has preceded the wide availability of chips. GCC and Glibc added support several years ago. But once all of the pieces come together what we'll essentially have is a significant step forward in our functional CPU architecture. The innovation of the stack, as a general-purpose catchall for dynamic data and control-flow was a huge innovation. But it always suffered from serious security vulnerability due to its flexibility and fragility. Adding an invisible control-flow-only shadow stack solves the problem elegantly.

An important issue found in OpenWRT package manager
The tech press has headlines like: "Patch now! Critical flaw found in OpenWRT router software"

The situation is not good, but it's not the end of the world. There's a potential supply chain exploit, and it's VERY GOOD that it was found and has now been fixed since it might conceivably have been exploited in targeted or wide-spread attacks.

The problem was discovered and responsibly disclosed earlier this year by Guido Vranken of a company called “ForAllSecure.” I think our listeners will find his discovery description very interesting:

https://blog.forallsecure.com/uncovering-openwrt-remote-code-execution-cve-2020-7982

For ForAllSecure, I’ve been focusing on finding bugs in OpenWRT using their Mayhem software. *[Mayhem, BTW, is a fuzzer.]* My research on OpenWRT has been a combination of writing custom harnesses, running binaries of the box without recompilation, and manual inspection of code.

I found this vulnerability initially by chance when I was preparing a Mayhem task for opkg. *[OPKG is the OpenWRT package manager.]* Mayhem can serve data either from a file or from a network socket. opkg downloads packages from downloads.openwrt.org, so my plan was to let this domain name point to 127.0.0.1 from which Mayhem is serving.

To test if opkg would indeed download packages from a custom network connection, I set up a local web server and created a file consisting of random bytes. When I ran opkg to install a package, it retrieved the file as I had intended, and then threw a segmentation fault.

I didn't understand why an invalid package would cause this error. After all, the package shouldn't be processed if the SHA256 hash was incorrect.

My initial hunch was that opkg would download the package, unpack it to a temporary directory, and only then verify the SHA256 hash before definitively installing it to the system. I suspected that the unpacker couldn't deal with malformed data, like the file with random bytes served from my web server.

Further inspection showed that the SHA256 hash wasn't checked at all, which is the basis of the vulnerability at hand.

I was right about the unpacker being buggy, though; malformed data would lead to a variety of memory violations.

Once I confirmed that opkg would attempt to unpack and install any package it downloads, I was able to recreate the findings with Mayhem with just a slight modification to opkg.

I set up a Mayhem task for opkg install attr (attr is a small OpenWRT package), and implicitly, Mayhem was able to find the remote code execution bug, by detecting the memory bugs in the package unpacker. If OpenWRT's SHA256 verification had worked as intended, opkg would simply discard the package and not process it, and no segmentation faults would transpire.

Mayhem is capable of fuzzing binaries without recompilation or instrumentation. Coming from a workflow that involves writing many custom harnesses for software libraries (which Mayhem also supports), this has been a delightful experience and it has allowed me to set up targets for dozens of OpenWRT applications in just weeks, and more vulnerability disclosures are forthcoming.

So, in other words, there are a bunch of problems with the OpenWRT codebase. But first and foremost is that its package manager is not bothering to check the SHA256 hash of anything it downloads. And, believe it or not, this problem is significantly compounded by the fact that these updates are over HTTP and not HTTPS. We know that means that there's no certificate to verify that the package is being obtained from the correct server. This makes a DNS spoofing or any other MITM-style traffic interception attack much easier to pull off.

So, the OpenWRT project recommends carefully upgrading to the latest version. That means doing what you can to make sure you are connecting to the correct server. Make sure that your DNS hasn't been changed, etc. Interestingly, there have been several router/DNS attacks recently. This might factor into that. The bug (CVE-2020-7982) was introduced in early 2017 and affects OpenWrt versions 18.06.0 through 18.06.6 and 19.07.0, and also separately the OpenWRT LEDE fork, version 17.01.0 through 17.01.7. The fix was applied to versions 18.06.7 and 19.07.1, released at the beginning of last month.
https://openwrt.org/advisory/2020-01-31-1

Cloudflare's 1.1.1.1 DNS is audited by KPMG
As we know, Mozilla's decision to route all of its browsers' DNS queries via DoH (DNS over HTTPS) to Cloudflare raised a bunch of noise when it was first announced. Many creaky old school UNIX diehards chafed over the loss of DNS's inherent distributed design. Perhaps some or much of this was due to their lack of long-term knowledge of who Cloudflare is. They may have thought that this was just some random provider. And, in truth, if Cloudflare was not as well known to me as they are, I'd have also been concerned by the idea of trusting some random single provider. But Cloudflare is not some random provider. We DO know Cloudflare and many of the people behind the name. So it always seemed like an excellent choice and a good idea to me.

That feeling has been solidified because Cloudflare opened their entire network to KPMG for the sake of allowing a fully independent audit. The auditor’s report has lots of detail, and I have a link to it in the show notes. But it concludes:
https://www.cloudflare.com/resources/assets/slt3lc6tev37/5xlHCvvNBrvrIoWbuk1vTy/e1058b0d366adf4e983aef99a6ed2a1f/Cloudflare_1.1.1.1_Public_Resolver_Report_-_03302020__2_.pdf

> “In our opinion, management’s assertion that the 1.1.1.1 Public DNS Resolver was effectively configured to support the achievement of Cloudflare’s Public Resolver commitments for the period from February 1, 2019 to October 31, 2019, based on the criteria above, is fairly stated, in all material respects.”

Closing The Loop
Many Tweets informed me that the new Edge Chromium browser is slated to be receiving vertical tabs. Yay!!

SQRL
SQRL now has a beautifully written, open source, cross-platform (Windows, Linux and MacOS) native SQRL client and library. It’s the work of Jose Gomex and Alex Hauser. Jose previously wrote the OAuth 2 provider for SQRL and Alex did a lot of work with Daniel's Android client. This client now has a forum on GRC’s SQRL forums at https://sqrl.grc.com and the project is being hosted on GitHub: https://github.com/sqrldev/SQRLDotNetClient

SpinRite
I was a bit slowed down by what I suspect was COVID-19. There’s increasing talk about a forthcoming antibody test, for which I’ll be first in line. I’m going to be very disappointed if I’m not positive for the antibody... so, fingers crossed.

But while I was unable to code I did manage to read the second of Hamilton’s “Salvation” trilogy... and it did not disappoint. I’m one book behind on Ryk Brown’s “Frontiers Saga.” I’ve read and very much enjoyed the first 27 books. So book number 28 awaits. But those are much lighter reading.

All that said, I was back to work on SpinRite’s AHCI driver this weekend and I’m hoping to have some new code to test soon.

Folding Proteins

We can donate our unused CPU cycles to help provide answers to COVID-19.

It's a happy coincidence that just as the SETI project decided that it didn't need any more raw signal processing number crunching, a crucial new need for distributed computing at massive scale would arise... because our unused CPU cycles can now help to increase our understanding of the structure and function of the COVID-19 virus.

Molecular modeling can be used to identify therapeutic drugs that might be of use in preventing the COVID-19 spike protein from binding to the ACE2 receptors of the cells in our lungs. In other words, our medical science and molecular modeling technology has progressed to the point that the field of "Computational Biology" is now a thing. It's possible to run simulations -- entirely in very sophisticated math -- to predict the 3D molecular shape and thus the interactions of biological compounds.

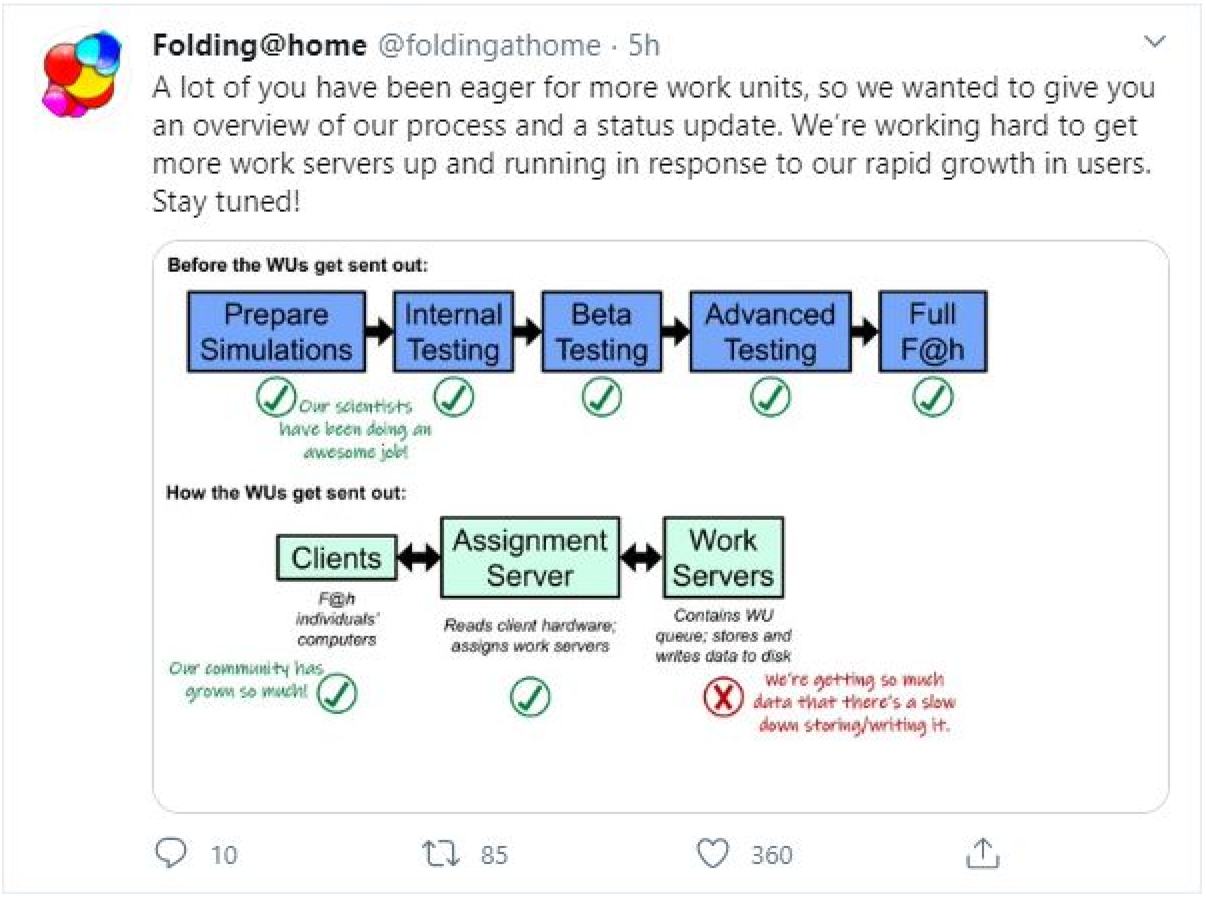
WikiPedia has a terrific fact-filled and up-to-the-minute introduction:

Folding@home (https://foldingathome.org) is a distributed computing project for performing molecular dynamics simulations of protein dynamics. Its initial focus was on protein folding but has shifted to more biomedical problems, such as Alzheimer's disease, cancer, COVID-19, and Ebola. The project uses the idle processing resources of personal computers owned by volunteers who have installed the software on their systems. Folding@home is currently based at the Washington University in St. Louis School of Medicine, under the directorship of Dr. Greg Bowman. The project was started by the Pande Laboratory at Stanford University, under the direction of Prof. Vijay Pande, who led the project until 2019. Since 2019, Folding@home has been led by Dr. Greg Bowman of Washington University in St. Louis, a former student of Dr. Pande.

The project has pioneered the utilization of central processing units (CPUs), graphics processing units (GPUs), PlayStation 3s, Message Passing Interface (used for computing on multi-core processors), and some Sony Xperia smartphones for distributed computing and scientific research. The project uses statistical simulation methodology that is a paradigm shift from traditional computing methods. As part of the client–server model network architecture, the volunteered machines each receive pieces of a simulation (work units), complete them, and return them to the project's database servers, where the units are compiled into an overall simulation. Volunteers can track their contributions on the Folding@home website, which makes volunteers' participation competitive and encourages long-term involvement.

Folding@home is one of the world's fastest computing systems. With heightened interest in the project as a result of the 2019–20 coronavirus pandemic, the system achieved a speed of approximately 768 petaFLOPS, or 1.5 x86 exaFLOPS, by March 25, 2020, making it the world's first exaFLOP computing system. This level of performance from its large-scale computing network has allowed researchers to run computationally costly atomic-level simulations of protein folding thousands of times longer than formerly achieved. Since its launch on 1 October 2000, the Pande Lab has produced 223 scientific research papers as a direct result of Folding@home. Results from the project's simulations agree well with experiments.

Okay… so, since February, the Folding@home community has been working on the computationally heavy work of figuring out how the COVID-19 virus proteins bind to cells. When the outbreak was picking up steam, the Folding@Home project asked for volunteers to donate their computers' unused computational power to help accelerate the open science effort to develop new life-saving therapies, as part of a collaboration of multiple laboratories around the world.



Folding@home @foldingathome · 5h

A lot of you have been eager for more work units, so we wanted to give you an overview of our process and a status update. We're working hard to get more work servers up and running in response to our rapid growth in users. Stay tuned!

Before the WUs get sent out:
Prepare Simulations
Internal Testing
Beta Testing
Advanced Testing
Full F@h
Our scientists have been doing an awesome job!
How the WUs get sent out:
Clients
F@h individuals' computers
Assignment Server
Reads client hardware; assigns work servers
Work Servers
Contains WU queue; stores and writes data to disk
Our community has grown so much!
We're getting so much data that there's a slow down storing/writing it.
10
85
360

Folding@Home says there's been a roughly 1,200% increase in contributors, with 400,000 new members signing up in the past two weeks. (And I have the feeling that they may see another upward jump once our listeners learn of this effort!) Due to the project basically being swamped recently, the Folding@home researchers are working to generate enough work for this massive influx of new processing power to tackle. They indicated that there might be a bit of downtime as new simulations are being set up. They wrote:

> Usually, your computer will never be idle, but we've had such an enthusiastic response to our COVID-19 work that you will see some intermittent downtime as we sprint to setup more simulations. Please be patient with us! There is a lot of valuable science to be done, and we're getting it running as quickly as we can.

So... https://foldingathome.org