Security Now! #894 - 10-25-22
Data Breach Responsibility

What are the chances??

This week on Security Now!

This week we note the release of an updated Firefox browser and Google's welcome and interesting announcement of a super-secure-by-design open source operating system project. We look at the latest cryptocurrency craziness and at a new Windows 0-day which bypasses downloaded executable file security checks. And speaking of 0-days, Apple just patched their iPhone and iPad OS's against their 9th 0-day of the year. We then take a look at the forces driving the evolutionary demise of previously rampant banking malware and at today's critical VMWare update. Then, after sharing and addressing some interesting listener feedback, we'll take a look at new Australian legislation aimed at punishing data breaches and consider the ethics of Australia's proposed new heavy fines.

Photo Credit:

Joel Rown🅰k @JoelRownak
Steve, I thought you'd get a kick out of this. There's supposed to be a parking meter there. I chose this parking spot because the meter was missing - a real money-saving opportunity. I noted that it was meter number 404. I don't need to tell you what that means in http!

Security News

Firefox 106 is out

Last Tuesday, Firefox 106 became publicly available with a collection of new features and security improvements. Feature-wise:

It is now possible to edit PDFs: including writing text, drawing, and adding signatures and setting Firefox as the default browser also makes it the default PDF application for Windows.

Private windows can now be pinned to the Windows taskbar in Win10 and 11 for simpler access and they have been redesigned to increase their feeling of privacy (they appear more dark and sparse.)

Swipe-to-navigate, with two fingers on a touchpad swiped left or right to perform history back or forward, now works for Linux users on Wayland. Wayland, for those unfamiliar with the term, is the successor to the original X Windows system.

Text Recognition in images allows users on macOS 10.15 and higher to extract text from the selected image (such as a meme or screenshot). The extracted text is copied to the clipboard in order to share, store, or search—without needing to manually retype.

Firefox's WebRTC capabilities also received a major upgrade with the upgrade of Firefox libwebrtc from version 86 to 103. That has brought a number of improvements:

- Better screen sharing for Windows and Linux Wayland users.
- Lower CPU usage and increased frame rates during WebRTC screen capture on macOS.
- RTP performance and reliability improvements.
- Richer statistics.
- Cross-browser and service compatibility improvements.

So, Firefox continues to move forward and to hopefully remain relevant for years to come.

Google's Open Source IoT KataOS and Sparrow

Driven by events and evidence, I have often bemoaned the increasingly sad state of affairs which describes the insecurity of today's operating systems and software. I've said that someday, somehow, this needs to change. So it was with some interest that I stumbled upon an announcement of just this sort of change last week which I wanted to share.

I'm not suggesting for a second that this effort by three talented software engineers at Google will ever amount to anything more than a learning experience for them and hopefully for others. But even if it never moves beyond GitHub, this is the sort of shape that an eventual effort will take. And I am quite clear about one thing: No closed commercial OS is ever going to get there. And all of the popular operating systems we're familiar with, Windows, Linux, Unix, macOS, were designed with security in mind, but in a fundamentally ad hoc fashion. They were all designed to have security **features**, but none were "secure-by-design" operating systems. So it's instructive to listen to the language these three developers use to describe their effort. It's the language of the future:

As we find ourselves increasingly surrounded by smart devices that collect and process information from their environment, it's more important now than ever that we have a simple solution to build verifiably secure systems for embedded hardware. If the devices around us can't be mathematically proven to keep data secure, then the personally-identifiable data they collect—such as images of people and recordings of their voices—could be accessible to malicious software.

Unfortunately, system security is often treated as a software feature that can be added to existing systems or solved with an extra piece of ASIC hardware— this generally is not good enough. Our team in Google Research has set out to solve this problem by building a provably secure platform that's optimized for embedded devices that run ML applications. This is an ongoing project with plenty left to do, but we're excited to share some early details and invite others to collaborate on the platform so we can all build intelligent ambient systems that have security built-in by default.

To begin collaborating with others, we've open sourced several components for our secure operating system, called KataOS, on GitHub, as well as partnered with Antmicro on their Renode simulator and related frameworks. As the foundation for this new operating system, we chose seL4 as the microkernel, because it puts security front and center; it is mathematically proven secure, with guaranteed confidentiality, integrity, and availability.

I'll just note that seL4 is an open source, verifiably secure microkernel with a heritage going back to the mid-90's. So this is exactly the right sort of foundation for a more functional OS to be built upon. They also use an seL4 framework known as CAmkES to provide statically-defined and analyzable system components. CAmkES stands for Component Architecture for micro kernel based Embedded Systems. It's a software development and runtime framework for quickly and reliably building microkernel-based multiserver (operating) systems. It follows a component-based software engineering approach to software design, resulting in a system that is modeled as a set of interacting software components. These software components have explicit interaction interfaces and a system design that explicitly details the connections between the components. This is the language of where we're headed. It's utterly different from the essentially ad hoc design of today's OSs where random coders grafted on new pieces and away it went. Continuing with the Google engineer's description of their work and offering:

KataOS provides a verifiably-secure platform that protects the user's privacy because it is logically impossible for applications to breach the kernel's hardware security protections and the system components are verifiably secure. KataOS is also implemented almost entirely in Rust, which provides a strong starting point for software security, since it eliminates entire classes of bugs, such as off-by-one errors and buffer overflows.

The current GitHub release includes most of the KataOS core pieces, including the frameworks we use for Rust (such as the sel4-sys crate, which provides seL4 syscall APIs), an alternate rootserver written in Rust (needed for dynamic system-wide memory management), and the kernel modifications to seL4 that can reclaim the memory used by the rootserver. And we've collaborated with Antmicro to enable GDB debugging and simulation for our target hardware with Renode.

Internally, KataOS also is able to dynamically load and run third-party applications built outside of the CAmkES framework. At the moment, the code on Github does not include the required components to run these applications, but we hope to publish these features in the near future.

To prove-out a secure ambient system in its entirety, we're also building a reference implementation for KataOS called Sparrow, which combines KataOS with a secured hardware platform. So in addition to the logically-secure operating system kernel, Sparrow includes a logically-secure root of trust built with OpenTitan on a RISC-V architecture. However, for our initial release, we're targeting a more standard 64-bit ARM platform running in simulation with QEMU.

Our goal is to open source all of Sparrow, including all hardware and software designs. For now, we're just getting started with an early release of KataOS on GitHub. So this is just the beginning, and we hope you will join us in building a future where intelligent ambient ML systems are always trustworthy.

https://opensource.googleblog.com/2022/10/announcing-kataos-and-sparrow.html

I wanted to share this because this is the sort of effort we're needing to begin to gain experience with the shape of an entirely new class of software which **won't** be fragile by design. We finally have sufficiently inexpensive processing power and memory so that design compromises won't be needed to make this sort of next generation system viable.

We've spoken before about how Windows' original designers wisely placed Windows GDI, its graphical device interface, outside of the Windows' kernel, up in userland, where its faults would not give attackers access to the kernel. But in their quest for more performance, Microsoft chose to discard security concerns and move GDI into the kernel. As we know, the result has been an apparently endless stream of serious vulnerabilities created because this extremely complex and error-prone code library was given full kernel privileges.

It's true that the ivory tower design of any operating system can be compromised. But we can hope that a system which has been built for the express purpose of exploring the possibility of creating truly secure code will never need to make such a mistake.

This Week in CryptoCurrency Craziness

This story revolves around an organization known as *"Mango Markets"* and in researching this I learned the ".markets" is now a top level domain. So Mango Markets is: https://mango.markets. And let me tell ya, it is a crazy place. Its self description is *"Decentralised, cross-margin trading up to 20x leverage with lightning speed and near-zero fees."* The top of its home page brags *"Long & short everything. Lightning fast · Near-zero fees · Permissionless"* Now, I strongly suspect that they didn't mean to be quite as "permissionless" as things turned out to be, because a serial DeFi-abusing hacker by the name of Avraham Eisenberg managed to take $14 million in cryptocurrency which he had retained from a previous DeFi hack and compound it into

$114 million from Mango Markets. Mango Markets' initial tweets immediately following the incident were:

Mango / @mangomarkets

We are currently investigating an incident where a hacker was able to drain funds from Mango via an oracle price manipulation. We are taking steps to have third parties freeze funds in flight. We will be disabling deposits on the front end as a precaution, and will keep you updated as the situation evolves. If you have any information, please contact blockworks@protonmail.com to discuss a bounty for the return of funds.

A short summary of what then transpired raised so many questions in my mind that I dug a bit deeper. Here's what the short summary I first encountered said:

Mango Markets exploiter comes forward: In a series of tweets over the weekend, an individual named Avraham Eisenberg took credit for the attack on Mango Markets following which he walked away with $114 million worth of cryptocurrency. Eisenberg came clean after he was publicly identified as the attacker last week and after the Mango Markets community voted to allow him to keep $47 million of the exploited funds if he returned $67 million back to the platform, so it and all the other projects that depended on it could avoid insolvency.

What has gone wrong with the world? Avraham was doxed and his identity became public knowledge, so he tweeted the following. Get a load of this:

Avraham Eisenberg / @avi_eisen

I was involved with a team that operated a highly profitable trading strategy last week.

I believe all of our actions were legal open market actions, using the protocol as designed, even if the development team did not fully anticipate all the consequences of setting parameters the way they are.

Unfortunately, the exchange this took place on, Mango Markets, became insolvent as a result, with the insurance fund being insufficient to cover all liquidations. This led to other users being unable to access their funds.

To remedy the situation, I helped negotiate a settlement agreement with the insurance fund with the goal of making all users whole as soon as possible as well as recapitalizing the exchange.

This is similar to how auto-deleveraging works on exchanges such as Binance and Bitmex, clawing back some profits from profitable traders in order to ensure all users' funds are protected.

As a result of this agreement, once the Mango team finishes processing, all users will be able to access their deposits in full with no loss of funds.

This wasn't a security breach or a hack in the sense of finding and exploiting a flaw in some code. This guy really knows his way around crypto-driven decentralized finance. Mango's first tweet mentions *"an oracle price manipulation"*. So we have some sort of half-cocked DeFi (you know, decentralized finance) system that was prone to manipulation. And this is the point I've been making here recently: Nothing about this entire wacky DeFi NFT world seems mature. That first note I encountered mentioned that this negotiated agreement was made by a vote of the stakeholders. I found that voting page and the so-called discussion is somewhat humorous. Many of the lesser stakeholders cannot imagine sanctioning this manipulation of the system: https://dao.mango.markets/dao/MNGO/proposal/GYhczJdNZAhG24dkkymWE9SUZv8xC4g8s9U8VF5Yprne. Interestingly, the voting strength is not per-member but by the size of one's wallet.

Anyway, this Avraham guy has a substack where he has talked about how all of this works and explained some of the schemes he's come up with for leveraging fluctuations in the DeFi world: Instead of DeFi it's "deepfivalue" https://deepfivalue.substack.com/. He currently has three long postings there:

Jul 10, 2021 :	Deep dives into cryptonomics
Aug 7, 2021 :	Anatomy of a Yield Farm
Jan 23, 2022:	How our team makes millions in crypto risk-free

Is "Jumped the Shark" the proper term for this whole crypto DeFi NFT blockchain mess?

New Windows 0-day bypasses executable security checks

Yesterday, BleepingComputer disclosed the news of a new 0-day flaw being exploited in the wild. The flaw enables JavaScript files downloaded over the Internet by attackers to run without producing any pop-up warnings to install the Magniber ransomware. And, it turns out that the flaw allows any executable to bypass Windows detection warnings.

We've talked about the Mark of The Web, MoTW. It's implemented using a very cool and underused feature of Microsoft's NTFS file system known as Alternate Data Streams. Alternate data streams have been a feature of NTFS from the start, though they've never received much attention. A file's alternate data stream is named and can be accessed by following the filename with a colon and the name of the alternate stream. It's just that simple. It's another dimension of file metadata. The Windows directory listing command "dir" can bee given the /R command line switch which will cause it to add any alternate data streams to its listing.

Whenever a file is downloaded over the Internet, it is automatically tagged as a less trusted file by assigning an alternate data stream named Zone.Identifier. This identifies the file as having originated from the scary Internet Zone and it causes several things to happen. One thing that happens is that the Windows SmartScreen system is invoked when anyone or anything attempts to run the file. SmartScreen raises an eyebrow and looks to see what it can determine about the file.

In addition, and this is where we were talking about the Mark of the Web earlier, Microsoft Office utilizes the MoTW flag to determine if the file should be opened in Protected View which causes macros to be disabled.

So, what has come to light is that there's a 0-day flaw, that's currently under active use, which has the effect of conveniently (for the bad guys) bypassing all of the Mark of The Web flagging and testing. HP's threat intelligence team reported that attackers are infecting devices with Magniber ransomware using JavaScript files.

And here's the kicker: The JavaScript files seen distributed by the Magniber threat actors are digitally signed but the signature is broken. Will Dormann, a senior vulnerability analyst at ANALYGENCE, discovered that the attackers signed these files with a deliberately malformed key. When any executable file is signed in this manner, even though it's been downloaded from the Internet and received a proper MoTW flag, Windows does not display any security warning, the SmartScreen check never occurs, and the file would automatically execute.

Will further tested the use of this malformed signature technique and was able to create his own proof-of-concept files that similarly bypass the MoTW and SmartScreen warnings.

Note that when an unsigned file is downloaded and opened in Windows 10, a MoTW security warning is properly displayed. And when a properly signed file is downloaded and opened in Windows 10, a MoTW security warning is properly displayed. It's only when a malformed signed file is downloaded and opened that Windows 10 remains silent and just runs the file.

This appears to be a bug that was introduced into Windows 10 since Will Dormann observed that a fully patched Windows 8.1 machine displays the MoTW security warning as expected. Also, something changed with Windows 11, since the bug does still have an effect when executing files from an archive, but not when running them directly.

It appears that the bug originated from the 'Check apps and files' SmartScreen feature under Windows Security > App & Browser Control > Reputation-based protection settings which first appeared in Windows 10. Disabling the "Check apps and files'' causes Windows to revert to its older behavior, where MotW prompts are unrelated to Authenticode signatures." It appears to be the signature checking that's resulting in a verification bypass. And, again, HP discovered this being used in the wild to sneak ransomware into Windows machines.

Finally, believe it or not, Will Dormann shared the proof-of-concept with Microsoft, who said they could not reproduce the MoTW security warning bypass... even though both he and Lawrence Abrams at BleepingComputer had easily done so. So... we'll see how this one goes.

Apple's 9th 0-day of the year bites the dust
For the record I'll note that yesterday, as part of a bundle of security updates, Apple patched and fixed its 9th iPhone 0-day vulnerability of the year. And Apple said the same thing they always say, which is that it “may have been actively exploited” although no one, even Apple, thinks that it wasn't. That's why it's a zero day.

All we know is that CVE-2022-42827 is an out-of-bounds write flaw reported to Apple by an anonymous researcher and caused by software writing data outside the boundaries of the current memory buffer. This can result in data corruption, application crashes, or, in the hands of an expert hacker, code execution. And Apple did say that if successfully exploited in attacks, this

0-day could have been used (oh yes, we're using the past tense already) by potential attackers to execute arbitrary code with kernel privileges.

The complete list of impacted devices includes iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, and iPad mini 5th generation and later. Apple addressed the zero-day vulnerability in iOS 16.1 and iPadOS 16 with improved bounds checking.

The evolutionary demise of banking malware

While assembling today's podcast I ran across some interesting commentary which described the various factors that have worked to reduce the prevalence of once dominant banking malware. I've edited it for our audience, and here's the gist of what happened:

Researchers from the security firm Mandiant have reported this week that URSNIF (aka Gozi, or Gozi/IFSB), one of the oldest and last few remaining banking trojan operations that were still active this year, has completely ditched its banking fraud-related features and now appears to operate as a basic backdoor trojan, the type of barebones malware typically used in Access-as-a-Service (AaaS) schemes that rent access to compromised devices. In other words, just creating an opening for others to use. According to Mandiant, the change took place this summer when URSNIF developers started distributing a new URSNIF version tracked under a codename of LDR4.

Mandiant cites several reasons for URSNIF's new radical redesign. At least two leaks of its earlier codebase, multiple branches of the same codebase that had slowly diverged and were making it harder to support features across different botnets, but also an ancient codebase that had finally reached the end of the road when IE was removed from Windows.

Mandiant said: “In June 2022, with Internet Explorer finally being fully removed from Microsoft Windows, the RM3 variant was officially seen as a "dead" malware from a technical point of view, as RM3 was reliant on IE for some of its critical network communication.”

It's a surprise that URSNIF lasted as long as it did operating on a banking trojan model. It had become obvious by the mid-2010s that the banking malware scene was dying, at least on the desktop. Banks, finally growing tired of a decade of thefts from customer accounts, at last rolled out multi-factor authentication and transaction verification systems. Though not foolproof, these systems did their job and made it more difficult and time-consuming for banking malware operators to steal money from compromised accounts.

Back in 2016, Emotet and TrickBot had converted their codebases from banking trojans to generic modular backdoors and were some of the first to do so. And even though they kept their banking modules around, Dridex and Qbot also followed suit later. In all instances, the primary driving force behind this shift in malware economics was the rise of ransomware and enterprise network big-game hunting. As it became clear to ransomware operators that they could extort obscene amounts of money from companies and government networks, they started to look for ways into those networks.

This led to the birth and rise of a market for initial access brokers where smaller threat actors would exploit corporate networking and server gear, plant backdoors, then sell access to these systems to ransomware gangs and their affiliates.

EvilCorp was the first major botnet operator to realize they could use their existing banking trojan to drop ransomware inside the thousands of corporate networks they had at their disposal through the Dridex botnet and even launched internal teams to write and deploy their own internal forms of ransomware.

Recall how I often noted that it was never safe to assume that an infected router would only be used as a proxy to reflect traffic or as a DDoS agent to flood. Sooner or later, the operators who had established a foothold on these routers were going to turn around and take a look inside the networks they had infected.

Because the Dridex botnet operated on a closed model where its operator providing limited access to their botnet to only a handful of very carefully vetted operators, the "service oriented" Emotet, and later TrickBot, cornered the market in Malware as a Service (MaaS) working with ransomware gangs.

And after law enforcement finally cracked down on Emotet and TrickBot, IcedID and Qbot stepped in as ready replacements after years of slowly growing in their shadows.

The world of underground malware is not difficult to understand. It's simply about the minimum amount of work that can be performed for the largest profit. Banking & carding is difficult, thanks to banks, and ransomware is easy, thanks to all the bazillion reasons we talk about every week.

There is no good reason to run a banking botnet these days, especially one as old and complicated as URSNIF. It's far easier to create and manage a simple backdoor, spam bored corporate employees until they infect themselves, and then sell access to ransomware or cryptomining gangs for a cut of the profits.

What this does mean, though, is that the urgency to remove anything that might manage to crawl into your network has never been greater. Stats from the latest intrusion reports indicate that ransomware can be deployed within 30 to 60 minutes of an initial intrusion. This strongly suggests that the deployment of responsive intrusion detection should be front of mind for all security teams.

VMWare's Critical CVSS 9.8 Update

Today, VMware released an update to repair a critical, low complexity, CVSS 9.8 remote code execution vulnerability in VMware Cloud Foundation which is their hybrid cloud platform for running enterprise apps in private or public environments.

This 9.8 out of 10 flaw, identified as CVE-2021-39144, was published August a year ago in 2021. The flaw exists in the open source XStream library which is used by Cloud Foundation. It carries that 9.8 rating because it can be exploited remotely by unauthenticated evildoers in low-complexity attacks that don't require user interaction. That's as bad as it can get. And due to the severity of the issue, which has existed for some time, VMware has also released security patches for other of their end-of-life products which remain in use.

VMWare's release advisory said: “Due to an unauthenticated endpoint that leverages XStream for input serialization in VMware Cloud Foundation (NSX-V), a malicious actor can get remote code execution in the context of 'root' on the appliance.”

The trouble is actually a deserialization flaw and we've discussed many times that data deserialization, which is inherently a data interpreter, is often a source of extremely subtle yet highly exploitable bugs. The National Vulnerability Database says: “XStream is a simple library to serialize objects to XML and back again. In affected versions, this vulnerability may allow a remote attacker who has sufficient rights to execute commands on the host only by manipulating the processed input stream.”

And speaking of flaws from last year, earlier this month VMware informed customers who updated to vCenter Server 8.0 (the latest version) that they'll have to keep waiting for a patch to address a high-severity privilege escalation vulnerability disclosed nearly a year ago, last November 2021.

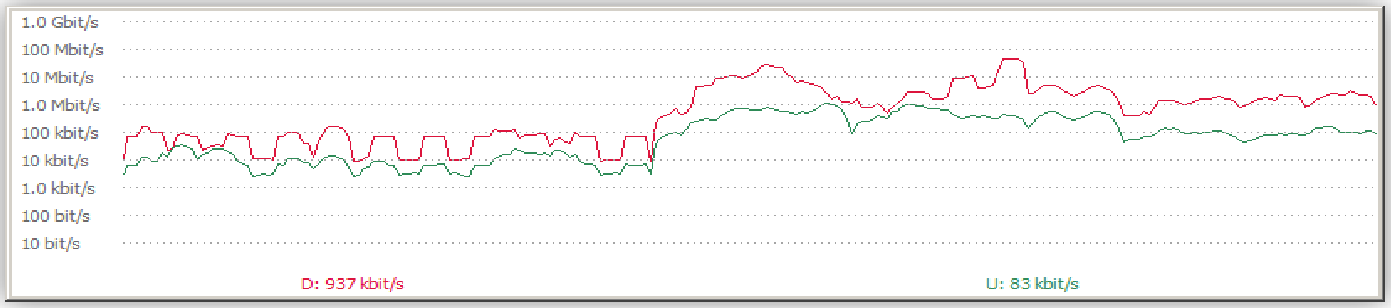
Closing The Loop

Nicolas Ross / @nicolasross

@SGgrc In SN884 you mentioned a piece of software you use to make graphs of snmp (that's what they are called) counters for you pfSense router. What was that software?

First of all, Nicolas was reminding me that SNMP – Simple Network Management Protocol – was the term that I blanked on during the podcast a few weeks ago when I was talking about monitoring the flow in and out of my pfSense-based router. The fabulous Freeware utility that I love for Windows, which I’ve talked about in the past, is called “Networx” by a company called SoftPerfect: https://www.softperfect.com/. The version I have is v5.5.5, which is the last of the freeware releases. After that it became $25 trialware for 30 days. In my opinion, it’s certainly worth $25, but v5.5.5, which is what I’m using and will continue to use, is available on the web. I just found it on FileHippo: https://filehippo.com/download_softperfect-networx/5.5.5/ and I have the link in the show notes. The download is digitally signed by a DigiCert certificate issued to SoftPerfect in 2016 using both SHA1 and SHA256. So double check for that and you can trust it.

The reason I love this so much is that although it has many features that may be of interest, three things stand out for me: First, as I mentioned at the time, it will monitor any interface’s SNMP counters. So I have it watching the WAN interface on my pfSense router. Second, it offers a logarithmic scale for bandwidth, which is so much better than a linear scale and it’s exactly what you want. That way you can see fluctuations in activity when not much is happening while still seeing maximum-rate connection-saturating transfers without going off scale. I have mine sized, as you can see in the screenshot below, for decade powers-of-10 scaling with the max set to one gigabyte. Since I can pull 300 megabits, that moves the graph into the last region.



So, whether you grab the freeware v5.5.5 or v7.x for $25, I recommend it without reservation. And their website reminded me that there's a version for macs, too.

Ando David Roots / @SQrooted

She: It's 3am. Why are you up?
Me: @SGgrc recommended a book
She: But you finished the Ryk Brown ones
Me: It's a new series, different author
She: 🫣

I'm on book #14 now and boy has it been an interesting ride so far. One of the most interesting topics raised across the series, and it's probably safe to say, though I'm not finished yet, that one of the series' primary explorations, is the truly intriguing question of the rights of sentient digital entities. Near the start of the first book we meet Julian, the sentient intelligence that runs a derelict starship. The series' primary human character, Alex, forms a deep bond with Julian and the books explore many aspects of digital intelligence.

Zacc / @ZacSixChip

@SGgrc Can we get a link to the hockey puck shaped power and reset button you mentioned on the show?

https://www.amazon.com/dp/B097QV6ZD2

Bart / @barcleyb

Hi Steve, I can't remember if it was the last SN episode or the one before, but I remember you mentioning how much faster Spinrite 6.1 can surface test todays larger drives. Was just wondering, is that your own tests or the feedback from the test community? I tried looking for a SR alpha/beta to experiment with on my own PCs/drives but nothing that would go past the drive detection phase (which by the way is working flawlessly with my particular setups). Have I missed something or is there no such release yet? - Will we be able to get past the dragons!?

The numbers I've been sharing are those from the testing community. It's great to hear that the drive detection phase worked well, since that's where the past year and a half have gone. Getting to that stage requires all of the new aspects of SpinRite to be working to such a degree that actually running SpinRite is almost anticlimactic. That said, none of the testing releases have gone any further, since it's that next step I'm currently working on – which is also happily also the last step. In those testing releases, SpinRite is already estimating the total time required to run on a drive by extrapolating from the speed tests it performs during its startup.

SKIN / @_SKIN_

Love your show! Question though… Why on earth is Spinrite 6.1 going to be free!? Not even an upgrade fee!? You're leaving money on the table. We want to support you! I think I'm going to buy it again just because of how hard you've worked on it.

The short version is "a promise is a promise", even if it's a promise I made nine years ago. And while that's enough, more than that, I'm sure that many people here have purchased SpinRite v6.0 based upon the promise that they would get v6.1 at no charge. And while this has turned out to be a much bigger challenge than I originally expected, I'm happy to do it because along the way I discovered some very cool new tricks that I didn't suspect a future SpinRite would be able to have up its sleeve. So there will be a SpinRite v7 and 7.1 and 7.2 and beyond. :)

Will Morhler @WillM

Last week's SN (the discussion of ECB mode of encryption) brought back a question I've always had: isn't any chaining mode of encryption brittle? If any part of the encrypted text is lost, isn't it the case that all subsequent blocks cannot be decrypted?

That's absolutely correct. But as we might say, that's more of a feature than a bug. One of the things we often want to do is to detect **any** tampering of any kind – deliberate or inadvertent – with a file's contents. Huge damage could be done by adding or removing a '0' from financial data. Since the integrity of the data is often just as important as its secrecy, we're often wishing to both encrypt **and** authenticate. And in the past we've spent some enjoyable time considering whether it's better to encrypt first then apply authentication to the encrypted result, or to authenticate a file before encrypting it. (Hint: if you cannot do both at once, and some cipher systems can and do, then you'll want to encrypt first and then apply authentication to the result. The reason is that upon reversing the process at the receiving end, you never want to decrypt anything that may have been altered. Always best to verify that the file has not been tampered with first and drop it immediately if it has been altered.

Getting back to the point, as a result of all that, having brittle encryption that will cause the file to essentially explode downstream of the point where anything was changed could be considered a good thing. :)

Ken Dudley / @dudley810

Hello what transcription software do you use for the podcast?

It's a fabulous system known as "Elaine Farris" at "edigitaltranscription.com"

Miscellany

For the first time ever I used Edge to assemble this document yesterday and today. The reason was that Chrome on my main Windows 7 system has become laggy for some reason and it's really become a problem with Google Docs, even just scrolling. I thought that it might be some extension I'm running, but I'm only running LastPass and uBlock Origin. I've been putting up with it for weeks, but yesterday the lag was really interfering with assembling this document.

So of course, I first switched to running Google Docs under Firefox. It was super snappy and ran like a charm, but the Verdana font I like to use had an annoying kerning problem. The letters were not spaced uniformly. It was conspicuous and I'm not the only person to have reported this. Many others online have posted looking for solutions. Nothing I tried worked. So I tried to ignore it but the wrongness of it bothered me.

Finally I thought, I'll use the other Chromium-based browser that I have, that being Edge. Well, Edge worked beautifully and I wrote and produced this entire show notes document in Edge.

So here's the thing that caused me to bring this all up: Under Edge and Windows 10, where I was working last evening, but not under Windows 7, the Windows' ALT-TAB key sequence switches among recently opened browser tabs in addition to top level desktop apps! As it happens, for me that's a HUGE win, since during the preparation of these notes I'm often jumping back and forth between browser tabs. Having seen this behavior under Windows 10 I'm now missing it today under Windows 7.

As all old Windows hands know, ALT-TAB is the Windows keyboard shortcut for changing among the desktop's foreground apps. And I'm often using it to quickly jump back and forth among running apps, typically cutting and pasting to move selected content around. But I've never seen it jump between different tabs on the same browser. It must be because Edge actually is running each browser tab in its own process space for inter-tab isolation. So in a sense the browser tabs really are individual browsers. But this was unexpected and I just wanted to make note of it.

I'll also note that Google just announced that Chrome's support for Windows 7 and 8.1 will be ending early next year. That's one of the dumbest things I've heard recently. Windows 7 still commands 10% of Windows desktops, one of mine being among them, and Windows 8.1 has an additional 2.7%. So that's 12.7%, or more than 1 out of every 8 desktops. And the dumbest thing about this is that it must be arbitrary and deliberate. There's absolutely no reason why Chrome should care whether it's running on top of Windows 7, 8.1, 10 or 11. If Chrome cannot ignore the version number of the operating system it's running on then they're doing it wrong.

Presumably, Google's decision to abandon Chrome on pre-Windows 10 machines synchronizes with the end of Microsoft's Extended Security Update (ESU) programs for Win7 and 8.1 on January 10th next year. I suppose that at some point I'll need to give up my use of Windows 7 on my primary working system... but I'd much rather just keep working.

Data Breach Responsibility

Over the weekend, a piece of surprising sort of over-the-top news from Australia got me thinking about the nature of breach culpability and legislative response. Everyone who's ben sharing this podcast for a few years will have heard and learned enough to have a well-formed and well-informed opinion about culpability. But not so our legislators. So, first, here's the new from Australia:

Australia: "Tougher penalties for serious data breaches"
On the heels of a month of high-profile and embarrassing attacks in Australia – Optus, Telstra, Medibank, Woolworths, and EnergyAustralia – three days ago on Saturday, the office of Australia's Attorney General posted an aggressive intention regarding legislation that is slated for discussion and adoption **this** week. One question that comes to mind is whether what happens in Australia remains in Australia? Or will this aggressive legislation prove to be the first of similar actions by other governments?

The Attorney General's office published:

The Albanese Government will next week introduce legislation to significantly increase penalties for repeated or serious privacy breaches. When Australians are asked to hand over their personal data they have a right to expect it will be protected. Unfortunately, significant privacy breaches in recent weeks have shown existing safeguards are inadequate.

It's not enough for a penalty for a major data breach to be seen as the cost of doing business. We need better laws to regulate how companies manage the huge amount of data they collect, and bigger penalties to incentivise better behavior.

The Privacy Legislation Amendment (Enforcement and Other Measures) Bill 2022 will increase maximum penalties that can be applied under the Privacy Act of 1988 for serious or repeated privacy breaches from the current $2.22 million penalty to whichever is the greater of:

- *$50 million;*
- *Three times the value of any benefit obtained through the misuse of information; or*
- *30 percent of a company's adjusted turnover in the relevant period.*

The Bill will also:

- *Provide the Australian Information Commissioner with greater powers to resolve privacy breaches;*
- *Strengthen the Notifiable Data Breaches scheme to ensure the Australian Information Commissioner has comprehensive knowledge and understanding of information compromised in a breach to assess the risk of harm to individuals; and*
- *Equip the Australian Information Commissioner and the Australian Communications and Media Authority with greater information sharing powers.*

This Bill is in addition to a comprehensive review of the Privacy Act by the Attorney-General's

Department that will be completed this year, with recommendations expected for further reform. I look forward to support from across the Parliament for this Bill, which is an essential part of the Government's agenda to ensure Australia's privacy framework is able to respond to new challenges in the digital era. The Albanese Government is committed to protecting Australians' personal information and to further strengthening privacy laws.

I can see both sides of this issue. For me, it's not as cut and dried as <quote> "Companies who hold consumer data, due to valid business needs, and who subsequently fail to keep such data private are so clearly to blame that they should be penalized with a crippling monetary fine."

In the real world we have this notion of "asking for it." Like, well, yeah, it's too bad that his car broke down when it did, but he hadn't taken it in for service in over ten years, so he was kind of asking for it. Right?

It seems to me that the underlying cause of a data breach **absolutely does** matter. Culpability need not be a mystery and it ought to be a factor. After all, it certainly is in all other aspects of the metering out of what we call "justice." That means that each issue needs to be taken on a case by case basis, just as it is elsewhere. In the case of data breaches, the legislature might not want to do that and may not know how to do that, but it's bad legislation if culpability isn't factored into the outcome.

If a company is shown to have been running a five-year old and many times obsoleted instance of VMware in the cloud, that was riddled with known and long since patched vulnerabilities, where one of them was inviting the bad guys into their network to root around and exfiltrate data, then, yes, any reasonable person would find their IT management grossly negligent, incompetent and culpable.

And on the subject of "asking for it" which relates to breach responsibility, I would add that when such a breach occurs, thus drawing everyone's attention to the fact and the nature of the breach, the leaked information, which would then be public, should be examined to see whether any personally damaging information was being retained in excess of the minimal needs of the business's purpose... because doing so should be a big no-no, too.

But if a company was running only the latest, currently-patched, state-of-the-art, updated software, active intrusion detection and network monitoring, holding periodic employee cyber-awareness conduct training classes, conducting simulated intrusion drills and randomly sending their own employees test bait phishing eMails... and yet, despite all of that, due to an entirely unknown 0-day vulnerability in the latest Microsoft enterprise software, someone crawled into their network and exfiltrated a database containing the bare minimum of personally identifiable customer information... how can it possibly be fair to levy a crippling fine, of no less than $50 million, onto such an enterprise?

The world as a whole is culpable for time and time again having chosen features over security; for having chosen to make our computer systems so overly complicated that they have become unknowable to anyone using them. We're all mostly just hoping that things are going to turn out alright.

This happened, of course, because it's not possible to sell **actual security** as a feature. Security is the absence of problems. You cannot sell an absence up front when problems take time to be found and revealed. Way back before the release of Windows XP, when Steve Ballmer was jumping up and down on stage bellowing at the top of his lungs that Windows XP was the most secure operating system that had ever been created, it wasn't possible to prove or to know then just how wrong time would prove that statement to be. I know Steve Ballmer and I like Steve Ballmer. He's a big huggable teddy bear. But as Bill Gates would say, he's not a "technical guy", so I imagine that Steve sincerely believed what he was selling. But it didn't matter how much he believed it, it wasn't true.

So we've built ourselves an industry where software is intellectual property that isn't sold, it's licensed. And where the agreement to license said software invariably incorporates a statement explaining that the licensor is making no representations of any kind whatsoever as to the operation or fitness of purpose of the software, and where the licensee agrees to hold the licensor harmless for any and all consequences of any failure on the part of the software to function as they hope. Often going so far, in the case of overly zealous attorneys, as to say that the software's publisher has no idea whatsoever whether the software works, what it does or what it may or may not do.

_____Sign here____

In such a world, while it is definitely possible for negligence to be the proximate cause of horrendously damaging data breaches, it is **also entirely possible** for a well meaning and perfectly well behaving company to be the blameless victim of the industry we have collectively built.

Until and unless things change, I dearly hope that all future legislation keeps this in mind. And I hope that any organization who did everything right, yet was breached anyway, is able to find a good law firm which is able to make this case for them.