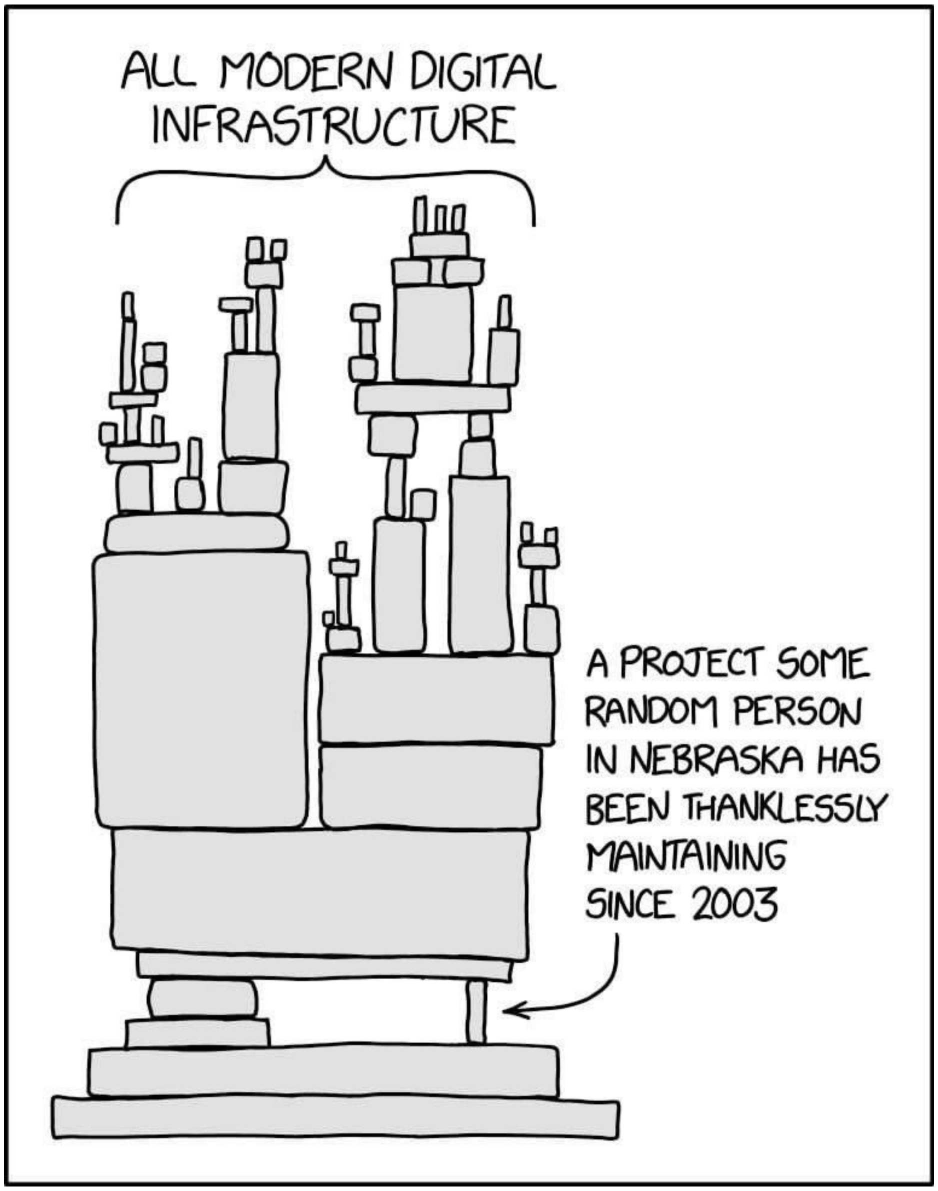
Security Now! #806 - 02-16-21
C.O.M.B.

This week on Security Now!

This week we'll begin by following-up on last week's headline making attack on the Oldsmar, Florida water treatment plant with news details that have since come to light. We'll then take a look into last week's Patch Tuesday event and at some of the sadly broken things that have once again been fixed. Also, anyone using Adobe's PDF tools, Acrobat or Reader needs to update. We're going to look at a dangerous Android App with 1.8 Billion (with a 'B') users, at Microsoft's note about the rise of web shells, which dovetails nicely into this week's WordPress addon disaster. I'll briefly update about my past eventful week with SpinRite, which includes a 25-second movie of new SpinRite code running. Then, we'll take a look at the recent discovery of the largest list of eMail & password combinations ever compiled... and what we can each do about it.


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Cyber-Event Followup

Oldsmar, Florida:
As I mentioned last week, I hoped the scare of the kinda-near-miss in Oldsmar, Florida attack might serve as a bit of a wake-up call, not only to other water treatment facilities, but also to the larger industrial control sector whose security pretty much must be wanting. Later last week something like that might have happened, at least in small part: The State of Massachusetts Department of Environmental Protection posted a "Cybersecurity Advisory for Public Water Suppliers — How public water suppliers can guard against cyber-attacks on water supplies."

This is a bit more specific that I was hoping for, since it must be that the security problems with industrial control extend into every area of the application of SCADA control systems. But the Massachusetts advisory did offer a few interesting new tidbits about the attack which occurred eleven days ago:

https://www.mass.gov/service-details/cybersecurity-advisory-for-public-water-suppliers

Dear Public Water Supplier,

We appreciate your attention to cybersecurity and the recent incident in Florida. Here is a more specific description on the events and suggested protective measures.

The FBI, DHS, US Secret Service, and the Pinellas County Sheriff's Office have issued a joint situational report that concerns the water sector. EPA is providing critical information from this report to the WSCC and GCC for awareness. EPA recommends that all water systems implement the mitigation measures listed at the end of this report where applicable.

On 5 February 2021, unidentified cyber actors obtained unauthorized access, on two separate occasions, approximately five hours apart, to the supervisory control and data acquisition (SCADA) system used at a local municipality's water treatment plant. The unidentified actors accessed the SCADA system's software and altered the amount of sodium hydroxide, a caustic chemical, used as part of the water treatment process. Water treatment plant personnel immediately noticed the change in dosing amounts and corrected the issue before the SCADA system's software detected the manipulation and alarmed due to the unauthorized change. As a result, the water treatment process remained unaffected and continued to operate as normal.

The unidentified actors accessed the water treatment plant's SCADA controls via remote access software, TeamViewer, which was installed on one of several computers the water treatment plant personnel used to conduct system status checks and to respond to alarms or any other issues that arose during the water treatment process. All computers used by water plant personnel were connected to the SCADA system and used the 32-bit version of the Windows 7 operating system. Further, all computers shared the same password for remote access and appeared to be connected directly to the Internet without any type of firewall protection installed.

Recommended Mitigation

- Restrict all remote connections to SCADA systems, specifically those that allow physical control and manipulation of devices within the SCADA network. One-way unidirectional monitoring devices are recommended to monitor SCADA systems remotely.

- Install a firewall software/hardware appliance with logging and ensure it is turned on. The firewall should be secluded and not permitted to communicate with unauthorized sources.

- Keep computers, devices, and applications, including SCADA/industrial control systems (ICS) software, patched and up-to-date.

- Use two-factor authentication with strong passwords.

- Only use secure networks and consider installing a virtual private network (VPN).

Implement an update- and patch-management cycle. Patch all systems for critical vulnerabilities, prioritizing timely patching of Internet-connected systems for known vulnerabilities and software processing Internet data, such as Web browsers, browser plugins, and document readers.

Everyone who listens to this podcast will have pretty much become security aware enough for none of that advice to be the least bit surprising. It does sound a little bit generic.

There HAS been some speculation about the timing of the attack within the security industry because a massive database containing 3.2 billion unique e-mail/password pairs was leaked online just three days before the attack. That database contained 13 entries relating to the Oldsmar facility, containing the domain "ci.oldsmar.fl.us". We'll be discussing the details and consequences of the publication of this massive list at the end of today's podcast.

But it would be a huge coincidence for the two to be related given that the newly leaked list contains 3.2 billion entries. I think that a more plausible explanation was offered by Christopher Krebs, the former and founding head of the US Cybersecurity and Infrastructure Security Agency (CISA). In Congressional testimony, Chris told a House of Representatives Homeland Security committee last Wednesday that the breach was "very likely" the work of "a disgruntled employee." I think that's extremely likely. This is not to excuse the obvious lax security throughout the installation.

One other note is that, especially since the pandemic, TeamViewer has become to remote control what Zoom has become for teleconferencing. But whereas the pressure of its public exposure, and of its very public security failures rapidly matured Zoom's security, TeamViewer has remained pretty much unchanged and those knowledgeable about TeamViewer are not impressed. Many of those in the Industrial Control Industry, who have been interviewed since the Oldsmar incident, have observed that they often see TeamViewer installed at sites and just shake their heads. Yeah, it works... but it's anything but secure.

Note that TeamViewer **does** offer multi-factor authentication. But you need to turn it on. And if the problem was a disgruntled employee, having everyone using the same credentials makes attribution difficult. And if it was an ex-employee, then all of the authentication would need to be changed following a termination. And, of course, that’s annoying. So it’s often overlooked.

Again, none of this is rocket science. Anyone listening to this pedant could easily tighten up the security at that location... and I’ll bet that everyone listening has better security for their home than Oldmar did for their water treatment plant.

Security News

Patch Tuesday

A lot happened with last Tuesday's patch batch. Overall, Microsoft addressed 56 security vulnerabilities — including 11 rated critical and six that were already publicly known. And, as expected, they finally lowered the boom on the use of unencrypted remote procedure calls to prevent the protocol fiddling that led to and enabled last year's Zerologon mess.

The 56 CVEs span the .NET Framework, Azure IoT, Azure Kubernetes Service, Microsoft Edge for Android, Exchange Server, Office and Office Services and Web Apps, Skype for Business and Lync, and Windows Defender.

The biggie that was found being exploited in the wild carries a surprisingly low severity rating of 7.8, making it just “important” but researchers noted that it deserves attention above some of the critical bugs in terms of patching priority.

It exists in Windows' Win32k operating system kernel and is an elevation-of-privilege (EoP) vulnerability. It would allow a logged-on user to execute code of their choosing with higher privileges, by running a specially crafted application. This would allow attackers to execute code in the context of the kernel and to operate with root SYSTEM privileges. And, again, it IS being used maliciously in the wild. So it's good that last week's updates will eliminate it.

There's also a publicly known critical flaw in the .NET Code and Visual Studio. But details are being closely held. Dustin Childs, of Trend Micro’s Zero Day Initiative said: “Without more information from Microsoft, that’s about all we know about it. Based on the CVSS severity scale, this could allow remote, unauthenticated attackers to execute arbitrary code on an affected system. Regardless, if you rely on the .NET Framework or .NET Core, make sure you test and deploy this one quickly.”

Believe it or not, there are also two CRITICAL-rated remote code execution flaws in Windows Fax Service. Fax Service? Does anyone Fax anything anymore? In any event, Microsoft said that an attacker who successfully exploited either vulnerability could take control of an affected system, and then be able to install programs; view, change or delete data; or create new accounts with full user rights. Wow. That sounds like one of those services you want to uncheck and remove from most, if not all, Windows systems.

Again, we always want to be minimizing our attack surfaces. We recently talked about turning off unneeded application layer gateways in our routers. Unneeded and unused features are inherently dangerous. If a system that has never needed to send or receive a fax doesn't have its fax service running or installed, even if its code is vulnerable it's not there to be attacked. In Windows' Control Panel on the Add or Remove Apps page, there's a link for adding or removing

Windows Features. You can browse that list and remove crap that Microsoft installed in order to minimize their tech support calls. They'd rather have the fax service running, taking up RAM, and making everyone vulnerable to the exploitation of its flaws, in the off chance that someone, somewhere, might someday need it ... than have that one person install it only if they know they need it.

The principle here is clear, whether it's an unneeded gateway in your router or a Windows' fax service you will never use: Services that are NOT running cannot hurt you. And without question, shutting them down or uninstalling them will reduce your attack profile. When you hear about systems being "security hardened" it's because all of this stuff has been turned off.

Now, let's turn to a service that no one can turn off or remove: Believe it or not, Windows is STILL experiencing remote code execution vulnerabilities in its TCP/IP stack. As we know, that's the last place you want to have such vulnerabilities because it is by definition exposed to the world. One flaw that Microsoft fixed last week was found in the way Windows handles IPv4 "source routing" and the other was in the way Windows handles IPv6 packet reassembly.

What was I just saying about the abuse of unused and unneeded code and services? Here's what Rapid7 has to say about IP Source Routing: "The host is configured to honor IP source routing options. Source routing is a feature of the IP protocol which allows the sender of a packet to specify which route the packet should take on the way to its destination (and on the way back). Source routing was originally designed to be used when a host did not have proper default routes in its routing table. However, source routing is rarely used for legitimate purposes nowadays. Attackers can abuse source routing to bypass firewalls or to map your network."
Oh, yeah! ... turn THAT on! Definitely!

Quoting Dustin Childs from Trend Micro's ZDI again, who was short and to the point about this one. He said: "IPv4 source routing should be disabled by default. You can also block source routing at firewalls or other perimeter devices." In other words, not today, nor for the past couple decades has anyone had problems with their routing tables. They work just fine, thank you very much. Notice that in the 15+ years of this podcast, where we have gone deep into the bit levels of the Internet, IP packets and routing, have we never even touched on IP Source Routing. It is **never** used in the real world. Yet, Microsoft just patched a CRITICAL flaw with a CVSS score of 9.8. So if, before last Tuesday, someone were to send your system a deliberately malformed IP packet containing source routing information, although the feature has never been used and is never needed, they could nevertheless take over your system remotely.

There really ought to be an install-time option for Windows. A big slider switch where you get to choose between "Security" on one hand and "Install a bunch of unneeded and never used extra features." It's got to be a choice because you really cannot have it both ways. I know I sound like I'm harping on Microsoft. But I'm very clear that anyone can make a mistake. You've never heard me jump on anyone for making a mistake. My entire problem surrounds policies. Organizations should be held to account for their policies. And Microsoft has a clearly demonstrated policy of installing unneeded services by default and supporting long-obsolete and unneeded protocols.

Which brings us to the other TCP/IP bug, the IPv6 packet reassembly bug. Packets are never supposed to become fragmented in transit from their sender to their receiver. It's inefficient and is assiduously avoided by all Internetworking equipment. It can occur when a router receives a large packet on one of its interfaces which needs to be sent out of another interface that has a lower MTU. MTU is "Maximum Transmission Unit." But much as with IP source routing, what made sense at the dawn of inter-networking has since largely become obsolete. Ethernet rules the world and the MTU of 1500 bytes has become the standard. There are so-called Jumbo Frames for Ethernet of 9000 bytes, with the idea of making Ethernet frames larger in order to reduce the per-packet addressing overhead. But such Jumbo frames are only useful within carefully controlled environments and they generally cause more trouble than they are worth.

Packet Fragmentation has turned out to be a long-standing annoyance within the Internet Protocol. Once a packet becomes fragmented it remains fragmented throughout the rest of its (or their) journey to their destination. And the packet reassembly problem turns out to be a particularly tricky wicket to code reliably. As a result, it has historically been a source of countless networking vulnerabilities. And so here we are again, in 2021, with Windows TCP/IP stack STILL not managing to get it right. And this creates another newly discovered CRITICAL vulnerability with a CVSS score of 9.8, enabling a full, unauthenticated, remote code execution vulnerability in Windows.

Because fragmentation is no longer supposed to occur, some high security firewalls simply drop all incoming fragmented packets. Or a router may choose not to fragment and forward an oversize packet, but rather to return an ICMP (Internet Control Message Protocol) packet explaining that the packet it has just received is too large to be forwarded. In which case the sender should reduce its transmitted packet size and try again. In other words, robust mechanisms exist for handling this without the need to ask the receiver to ever reassemble packet fragments.

Last Tuesday also fixed a critical bug in the Windows Camera Codec Pack, and another in the Windows DNS server. Thankfully, that is one service that Microsoft doesn't install by default. On the other hand, it's a highly critical remote code execution bug if the service is running and receives a maliciously formed DNS query. A critical flaw was also fixed in Windows' print spooler, in the .NET Core for Linux, and in the Windows Codecs Library.

So... pretty much just your typical month of the life of Windows 10. They are never going to get it right because their economic model requires that they keep fussing with it. And all experience shows that every time you touch it you risk breaking something that used to work.

Adobe Acrobat & Reader

Last Tuesday Adobe released critical updates to three versions each of its Acrobat and Reader products. They are all subject to targeted remote code execution attacks. Since opening a PDF is generally regarded as safe, the attacks might be expected to escalate now that the fix against them is being made widely available.

So, if you're a user of Acrobat DC, 2020 or 2017, or Reader DC, 2020 or 2017 on either Windows of macOS, it would be a good idea to ask your installed software to check for updates to itself and move to the latest release.

Android SHAREit
I don't normally talk about Android App security flaws and problems, I suppose because it seems like such low-hanging fruit. I mean, our time here is limited every week. But, when vulnerabilities have been found and responsibly disclosed in an Android App that has 1.8 Billion users worldwide, and when after 90 days of no response from that app's publisher the responsible disclosing party decides to finally go public with their vulnerability information... then the situation rises to the level that we need to touch on it here.

The Android app in question is SHAREit, and even I had heard of it. And I'd venture that many of our Android-equipped listeners have a copy. The Google Play store lists it as: “SHAREit - Transfer & Share” It is a super-popular file exchange app.

https://www.trendmicro.com/en_us/research/21/b/shareit-flaw-could-lead-to-remote-code-execution.html

It was Trend Micro who examined, discovered and reported the problem to SHAREit's publisher, and received no feedback. Trend Micro wrote:

“We discovered several vulnerabilities in the application named SHAREit. The vulnerabilities can be abused to leak a user’s sensitive data and execute arbitrary code with SHAREit permissions by using a malicious code or app. They can also potentially lead to Remote Code Execution (RCE). In the past, vulnerabilities that can be used to download and steal files from users’ devices have also been associated with the app. While the app allows the transfer and download of various file types, such as Android Package (APK), the vulnerabilities related to these features are most likely unintended flaws.”

SHAREit has over 1 billion downloads in Google Play and has been named as one of the most downloaded applications in 2019. Google has been informed of these vulnerabilities.

Trend Micro's posting then delves into the details of the App's API registrations and operation which allow arbitrary files to be downloaded and executed, including downloading and installing any APK. It's not that the SHAREit app itself is malicious, but that its presence in all 1.8 Billion Android devices creates security holes large enough to drive Google through. So this creates a huge opportunity for abuse by any other app that wants to get up to some mischief. And now, after Trend Micro's full disclosure, everyone knows about it.

Trend Micro concluded their detailed write-up saying:

“We reported these vulnerabilities to the vendor, who has not responded yet. We decided to disclose our research three months after reporting this, since many users might be affected by this attack because the attacker can steal sensitive data and do anything with the apps’ permission. It is also not easily detectable.”

So, just a heads-up for any users of SHAREit. If it's one of those free Android apps that you once downloaded, used for a while, then forgot about... but it's still installed... now would be a good time to say goodbye. If you use it regularly and cannot live without it, then perhaps be on the lookout for its publisher to finally take the security of their app seriously with an update. Or perhaps find a more responsibly-written alternative.

The Rise of The Web Shells
Last Thursday, Microsoft's Detection and Response Team (DART), posted an interesting update about a rapidly growing security threat we've never discussed: “Web Shells.” So first we need to define a web shell. It's a fancy name for some malicious code that you definitely don't want running on your server:

A web shell is typically a small bit of malicious code, written in whatever typical server-side web development programming language your server supports. That would be something like ASP (Active Server Pages), PHP (Personal Home Page), JSP (Java Server Pages), or similar. Attacker somehow arrange to implant this script onto a victim's web server to then provide long-term persistent remote access and code execution of server functions. These web shells allow attackers to run commands on servers to steal data or use the server as a launch pad for other activities like credential theft, lateral movement, deployment of additional payloads, or hands-on-keyboard activity... and most significantly they allow attackers to rather trivially maintain a persistent presence within an affected organization.

This gets picked up by our radar because Microsoft has been monitoring this as a rapidly growing trend. They wrote:

“One year ago, we reported the steady increase in the use of web shells in attacks worldwide. The latest Microsoft 365 Defender data shows that this trend not only continued, it accelerated: every month from August 2020 to January 2021, we registered an average of 140,000 encounters of these threats on servers, almost double the 77,000 monthly average we saw [over the same period, year over year compared to] last year. The escalating prevalence of web shells may be attributed to how simple and effective they can be for attackers. As web shells are increasingly more common in attacks, both commodity and targeted, we continue to monitor and investigate this trend to ensure customers are protected.”

Microsoft's blog posting provides some riveting case history:

Attackers install web shells on servers by taking advantage of security gaps, typically vulnerabilities in web applications, in internet-facing servers. [Can you say “WordPress add-ons”?] These attackers scan the internet, often using public scanning interfaces like shodan.io, to locate servers to target. They may use previously fixed vulnerabilities that unfortunately remain unpatched in many servers, but they are also known to quickly take advantage of newly disclosed vulnerabilities.

For example, last June 30, F5 Networks released a patch for CVE-2020-5902, a remote code execution (RCE) vulnerability in Traffic Management User Interface (TMUI). The vulnerability is a directory traversal bug with a CVSS score of 9.8. Just four days later, on July 4, exploit code was added to a Metasploit module.

The following day, Microsoft researchers started seeing the exploit being used by attackers to upload a web shell to vulnerable servers. The web shell was used to run common cryptocurrency miners. In the days that followed, industry security researchers saw the exploit being broadly used to deploy web shells, with multiple variants surfacing not long after.

Once installed on a server, web shells serve as one of the most effective means of persistence in an enterprise. We frequently see cases where web shells are used solely as a persistence mechanism. Web shells guarantee that a backdoor exists in a compromised network, because an attacker leaves a malicious implant after establishing an initial foothold on a server. If left undetected, web shells provide a way for attackers to continue to gather data from and monetize the networks that they have access to.

Okay... so just to make the mechanism clear: a web shell script is just a static text file with an extension like .asp, .php, .jsp and so on. The typical web server will have a bazillion such files. They're little fragments of the whole system which are invoked on demand. And if one additional .asp or .php file were to be added among hundred of others, who would ever know? No one who has installed any of today's web-based systems has any idea what all of that crap is, anyway. So an additional file would go completely unnoticed.

Then, at any time (or times) later, a remote attacker simply queries a specific URL on that company's web server which references the web script that was deposited into an script executable directory, and the attacker's code comes to life. It's really kind of diabolical. It's trivial and doesn't take a rocket scientist. After all, PHP was designed to be a dead simple web page scripting authoring language. You just always hope that it's only YOUR code that your server is running.

What Microsoft points out is that the instant a new vulnerability surfaces, attackers are scanning the Internet and using the new vulnerability to drop web shells onto briefly-vulnerable targets. And once there, the script can sit unseen. If minutes later the vulnerability is patched to foreclose on any further use, it's too late. Now that server is carrying a malicious script that will jump into service after being invoked through the simple receipt of a querying URL.

Microsoft notes that the challenge of discovery of such implants is hampered by the sheer volume of network traffic plus the usual noise of constant internet attacks. This means that targeted traffic aimed at a web server can blend right in, making detection of web shells a lot harder and requiring advanced behavior-based detections that can identify and stop malicious activities that hide in plain sight.

And it's also possible to chain attacks. Microsoft explains:

Another challenge in detecting web shells is uncovering intent. A harmless-seeming script can be malicious depending on intent. But when attackers can upload arbitrary input files in the web directory, then they can upload a full-featured web shell that allows arbitrary code execution—which some very simple web shells do.

These file-upload web shells are simple, lightweight, and easily overlooked because they cannot execute attacker commands on their own. Instead, they can only upload files, such as full-featured web shells, onto web servers. Because of their simplicity, they are difficult to detect and can be dismissed as benign, and so they are often used by attackers for persistence or for early stages of exploitation.

Finally, attackers are known to hide web shells in non-executable file formats, such as media files. Web servers configured to execute server-side code create additional challenges for detecting web shells, because on a web server, a media file is scanned for server-side execution *instructions. Attackers can hide web shell scripts within a photo and upload it to a web server. When this file is loaded and analyzed on a workstation, the photo is harmless. But when a web browser asks a server for this file, malicious code executes server side.*

So again, as we often see, we have a double-edged sword. Empowering web servers to interpret textual scripts as code is incredibly empowering. In fact, this method of running server side code has pretty much taken over the world. The down side is that ... uhhhh ... text files on the server are being interpreted as code. In other words, it's both what we want and what we don't want. And it's a growing problem.

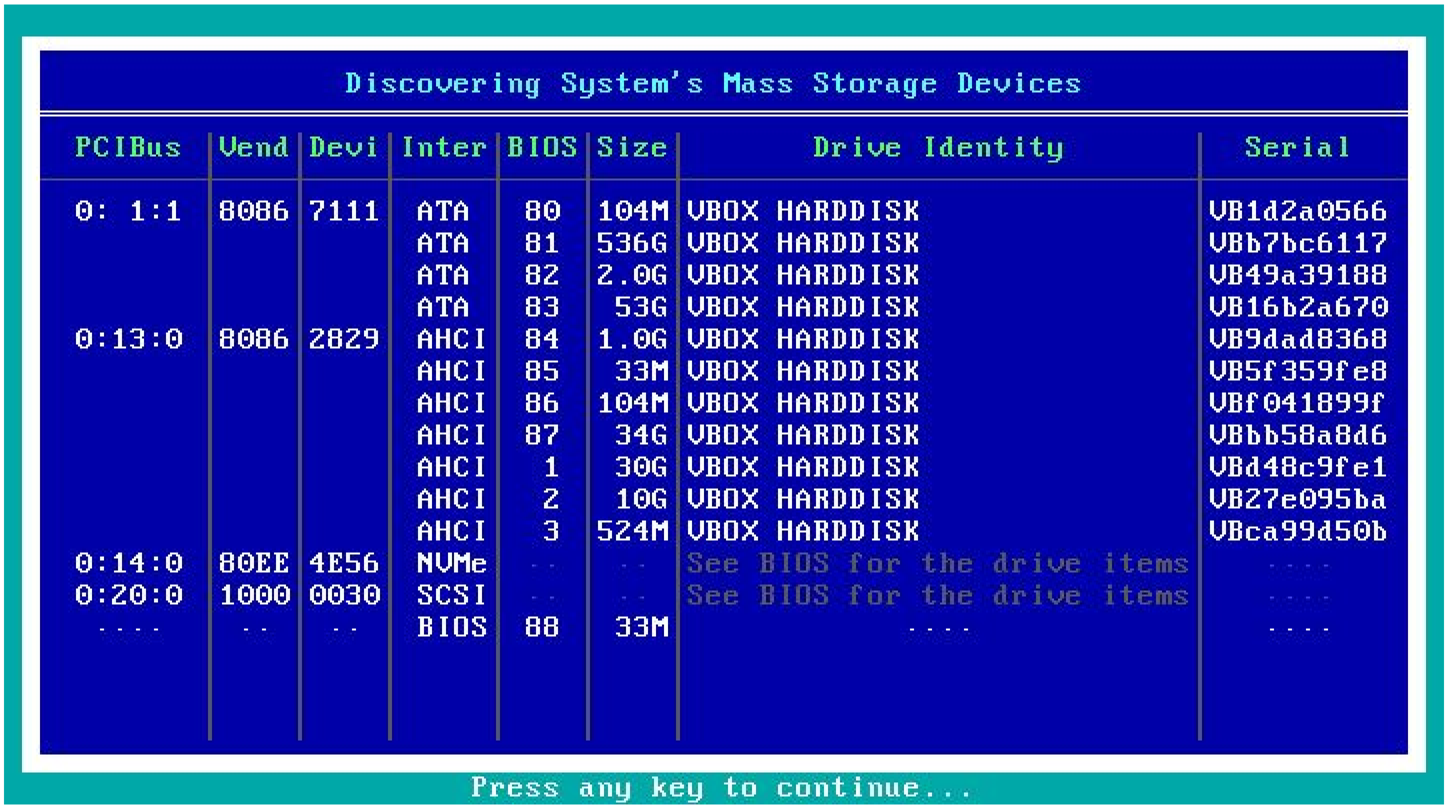
This week's WordPress Mess
The "Responsive Menu" WordPress plugin is exposing more than 100,000 WordPress sites to multiple critical and high severity vulnerabilities. Responsive Menu is a WordPress plugin designed to help admins create W3C compliant and mobile-ready site menus.

The Wordfence Threat Intelligence folks found three vulnerabilities that can be exploited by attackers with basic user permissions to upload arbitrary files and remotely execute arbitrary code. What was that I was just saying about the rise of Web Shells? The first of the three flaws enables authenticated attackers to upload arbitrary files which eventually allows them to achieve remote code execution. The other two vulnerabilities allow an attacker to forge requests to modify plugin settings of the plugin which, in turn, allows them to upload arbitrary files allowing for remote code execution. To leverage these vulnerabilities, attackers logged in as subscribers, or another low-level user, upload menu themes archived as ZIP files and containing malicious PHP files. Once the archive is extracted for installation, the attacker can access the files via the site frontend to remotely execute the malicious code which ultimately can lead to a full site takeover.

This sounds like an involved and complex dance to go through. But if the target has sufficient value, and if, for example, a persistent web shell can be dropped into a briefly vulnerable server, then it might be worth the effort.

SpinRite

Last week I talked about the anxiety-provoking "Discovering System's Mass Storage Devices" screen where, if SpinRite was going to have a problem, that was where it would happen while it was scanning the system's drives. So I decided to post SpinRite's process and progress as it went along. At this time last week that was just a concept. Today it exists. I finished that work and posted a sample image of the new SpinRite screen to my "work progress tracking" thread of my blog on the GRC forums, and also to GRC's spinrite.dev newsgroup:



```
                 Discovering System's Mass Storage Devices

 PCIBus |Vend|Devi|Inter|BIOS|Size|       Drive Identity        |  Serial

 0: 1:1  8086 7111  ATA   80  104M VBOX HARDDISK                 VB1d2a0566
                    ATA   81  536G VBOX HARDDISK                 VBb7bc6117
                    ATA   82  2.0G VBOX HARDDISK                 VB49a39188
                    ATA   83   53G VBOX HARDDISK                 VB16b2a670
 0:13:0  8086 2829  AHCI  84  1.0G VBOX HARDDISK                 VB9dad8368
                    AHCI  85   33M VBOX HARDDISK                 VB5f359fe8
                    AHCI  86  104M VBOX HARDDISK                 VBf041899f
                    AHCI  87   34G VBOX HARDDISK                 VBbb58a8d6
                    AHCI   1   30G VBOX HARDDISK                 VBd48c9fe1
                    AHCI   2   10G VBOX HARDDISK                 VB27e095ba
                    AHCI   3  524M VBOX HARDDISK                 VBca99d50b
 0:14:0  80EE 4E56  NVMe  --    -- See BIOS for the drive items     ----
 0:20:0  1000 0030  SCSI  --    -- See BIOS for the drive items     ----
  ----    --   --   BIOS  88   33M            ----                  ----

                       Press any key to continue...
```

https://www.grc.com/sr/Discovering_System's_Mass_Storage_Devices_2.png

Because it's sort of fun to see the actually operating, I created a short 25-second video which captures SpinRite running on my mega torture-test SpinRite development machine. It's our grc.sc shortcut of the week, so anyone can view it with https://grc.sc/806

(http://media.grc.com/mp4/Discovering-SRv6.1.mp4)

At the end of last week I released the first test of this new beginning startup phase of SpinRite for the newsgroup gang to pound on and test. Overall, it went very well. There was a one-off issue that had plagued me back with ReadSpeed with one of our testers in Germany. He has a Gigabyte motherboard with an AMD Phenom processor. If he boots from diskette, no problem. But if he boots from USB the system hangs. I managed to fix it once under ReadSpeed, but the trouble has returned. So I found and have purchased that motherboard from eBay. It's a very subtle hang and I need to get my hands on it to see what's going on.

But most people reported boredom because everything worked perfectly for them. But those boring positive results go a long way to establishing that the code is working in every system we can throw at it. There were three systems where the physical-to-logical drive associator was unable to match-up a hardware drive to its BIOS designation. This means that the same drive would appear twice in SpinRite's drive list — once as a BIOS-accessed drive and then also as the one we want: the new direct hardware-access drive. So I have a bit more work to do on the logical-to-physical drive associator. And once I have that nailed down I'll move on to get the drive list and selection screen updated with all of the new drive and controller information that will be available.

I also ordered and received the first component of the OnTime RTOS-32 embedded OS platform that I'll be porting SpinRite v6.1 over to, after v6.1 is released. That will create the first 32-bit SpinRite as v7.0 allowing SpinRite to boot, for the first time ever, from either a BIOS or a UEFI-based system. And at that point I'll then move forward to add hardware-level USB support into v7.1 and hardware NVMe support into v7.2. And then continue moving forward adding feature after feature.

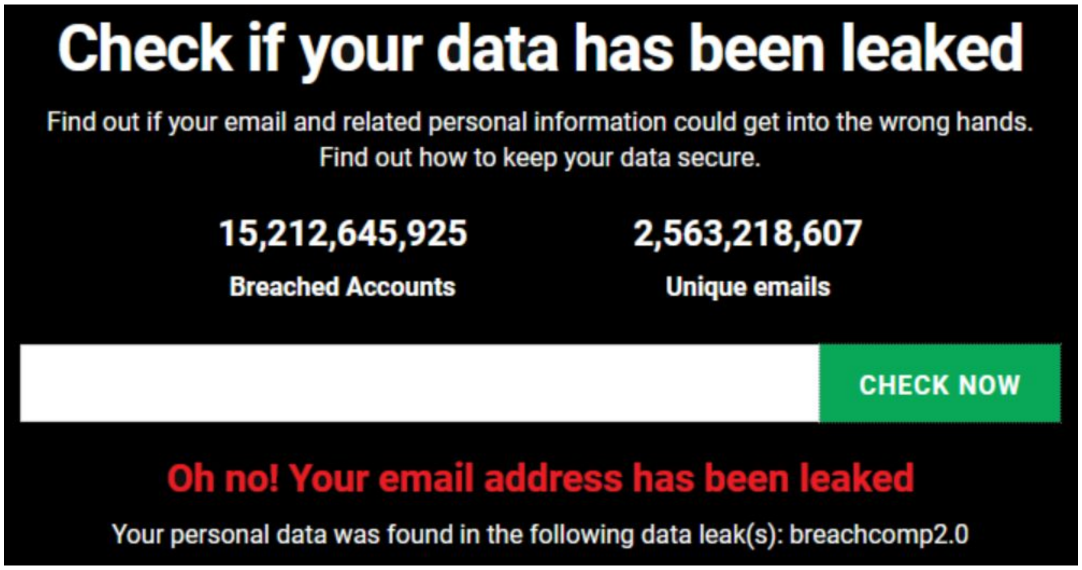
I haven't even looked at or installed the new OS platform. Work on SpinRite v6.1 comes first. But at some point, when I'm waiting for feedback on a 6.1 test release, I plan to get my feet wet with the new OS platform by writing a quick "Hello World!" app for it, just to positively verify that we have something that will dual-boot and be able to host SpinRite 7.0 and beyond.

So, go to https://grc.sc/806 and checkout SpinRite's new drive discovery video. It's fun to watch.

C.O.M.B.

Compilation Of Many Breaches

Whoops! My own personal gMail eMail has been leaked!



Check if your data has been leaked

Find out if your email and related personal information could get into the wrong hands.
Find out how to keep your data secure.

15,212,645,925 Breached Accounts

2,563,218,607 Unique emails

Oh no! Your email address has been leaked

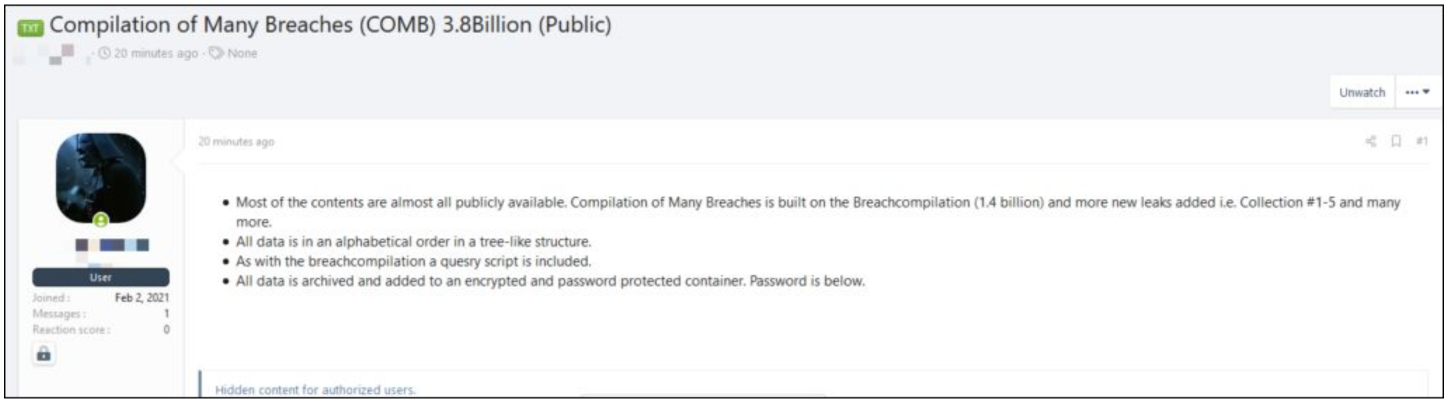
Your personal data was found in the following data leak(s): breachcomp2.0

https://cybernews.com/personal-data-leak-check/
(https://grc.sc/comb)

The Cybernews folks wrote: "It's being called the biggest breach of all time and the mother of all breaches." Of course, the mother of all breaches would be MOAB, which would be a **far** cooler name, though technically MOAB has already been taken to be the "Mother Of All Bombs.")

Anyway, we haven't checked-in on the status of the underground dark web's aggregation and maintenance of massive login credential lists in quite some time. So I thought that the recent appearance of the latest and — by any measure — greatest, such compilation would be a good opportunity to see where all that currently stands.

COMB - the Compilation of Many Breaches - contains more than 3.2 Billion (with a 'B') unique pairs of cleartext emails with their matching passwords. I've embedded a screenshot of the hacker's February 2nd announcement posting in the show notes:


TXT Compilation of Many Breaches (COMB) 3.8Billion (Public)

[redacted] · 20 minutes ago · None

Unwatch

[redacted]

User

Joined: Feb 2, 2021
Messages: 1
Reaction score: 0

20 minutes ago #1

- Most of the contents are almost all publicly available. Compilation of Many Breaches is built on the Breachcompilation (1.4 billion) and more new leaks added i.e. Collection #1-5 and many more.
- All data is in an alphabetical order in a tree-like structure.
- As with the breachcompilation a quesry script is included.
- All data is archived and added to an encrypted and password protected container. Password is below.

Hidden content for authorized users.

The posting contains four bullet points highlight the features of this beast:

- Most of the contents are almost all publicly available. "Compilation of Many Breaches" is built on the "Breach Compilation" (1.4 billion) and more new leaks added I.e. Collection #1-5 and many more.
- All data is in an alphabetical order in a tree-like structure.
- As with the Breachcompilation, a query script is included.
- All data is archived and added to an encrypted and password protected container. Password is below.

CyberNews claims that they were the first leak test database to include the COMB data, and since COMB was first released they said that nearly 1 million users had used their personal data leak checker to see if their data was included within this the biggest breach compilation of all time. I checked all of my various GRC eMail addresses and they were all clean. But when I checked my gMail account, which I use as a generic spam-filtering bit bucket, I received the screen above informing me that my eMail address had appeared in a breach.

Fortunately, I do change that password from time to time, it's long total entropy gibberish, and I have multi-factor authentication enabled — less for gMail than for my Google account. But, I never use that gMail account for anything that might need password recovery — so nothing sensitive is even passing through — and although it's unusual behavior with gMail, I routinely delete everything in my inbox and trash, including all archival storage. So IF, in the unlikely case, someone were to gain access, they wouldn't find much of interest.

But, with any database this massive it's probably useful to see whether you're represented there. So I created a grc.sc shortcut link to their checker: https://grc.sc/comb

So, what do we know about this massive compilation?

Exactly two weeks ago, on February 2, COMB was leaked on a popular hacking forum. It contains billions (literally) of user credentials lovingly compiled from many past breaches of Netflix, LinkedIn, Exploit.in, Bitcoin and more. Is similar in character to the Breach Compilation of 2017, which leaked 1.4 billion credentials. The difference being that COMB more than doubles the number of unique eMail and password pairs.

As the individual announcing the database notes, it includes a script named count_total.sh, which was also included in the 2017 Breach Compilation. The breach also includes query.sh, for querying emails, and sorter.sh for sorting the data. Running "count_total.sh" — a straightforward bash script to count and sum the number of lines in each of the files, returns 3.27 billion email/password pairs.

Cybernews noted that it was neither clear nor specified which previously leaked databases were collected in this latest COMB — but the presumption is "all of them" and a perusal of the list makes it clear that there are email addresses and passwords for domains around the world. Because COMB is a quick, searchable, well-organized database of (probably all) past major leaks, you would expect to find, and you do, addresses from Netflix, Gmail, Hotmail, Yahoo and more.
The Cybernews guys did some counting and found approximately 200 million Gmail addresses and 450 million Yahoo email addresses. They also noted that neither Netflix nor Gmail have ever publicly announced a known breach of their user data. But the reuse of passwords, which was hopefully far more frequent in the past than now, meant that breaches of other sites could result in the disclosure of shared passwords.

On the flip side, Microsoft did confirm that between January and March 2019, hackers were able to access a number of consumer Outlook.com, Hotmail and MSN Mail email accounts. And we know that the mother of all breaches apparently suffered back in 2014 — though it was kept quiet for two years until 2016 — by Yahoo, who finally confirmed what many in the security industry had deduced through its effect, that all 3 billion of its users' accounts had been impacted by a breach.

It appears that not all of the data from past Yahoo and Hotmail/Microsoft breaches have been included in COMB. Though hackers typically boast about the size of their — uh — credential lists, there is also some pruning and cleaning of knowably-dead credentials. Even so, the leak lists keep getting bigger. This COMB list clearly builds upon 2017's "Breach Compilation" list with its 1.4 billion email/password pairs, all in plaintext. And that one was nearly twice the size of the previous credential exposure from "Exploit.in" which consisted of nearly 800 million records.

We know that the 2017's Breach Compilation was an amalgamation of 252 previous breaches, including those aggregated from the previous "Anti Public" and Exploit.in dumps, as well as LinkedIn, Netflix, Minecraft, Badoo, Bitcoin and Pastebin.

These days, many exfiltrated passwords are now hashed, older one's weren't, and in the early days of password hashing, those that were, were not well hashed — meaning few, if any, iterations and often no salt added, let alone per-user salt. Consequently, some raw databases contain eMail in plaintext and passwords as hashes. However, when the data from the previous Breach Compilation was analyzed, it was found that "14% of exposed username/passwords pairs had not previously been decrypted by the community and were now available in clear text." So, reversing password hashes is an ongoing campaign.

And, of course, there's the question of age. The older the lists the less useful they are. However, when the Identity Intelligence company 4iQ discovered the Breach Compilation in 2017, they tested a small subset of the passwords for verification, and most of the tested passwords worked. The intelligence analysts stated that they found the 41GB dump on December 5, 2017, with the latest data updated on November 29, 2017. (We talked about this at the time, so that might ring some bells.)

The 4iQ researchers note that the leak was not just a list, but rather an "interactive database" that allowed for "fast (one second response) searches and new breach imports. Given the fact that people reuse passwords across their email, social media, e-commerce, banking and work accounts, hackers can automate account hijacking or account takeover."

So what are our takeaways from the news of this massive list?

For one thing, old, forgotten and unused accounts often remain active. Something somewhere that you might not have used for years might still be there and might allow a bad guy to gain a foothold. So, ask yourself about any accounts that you haven't used for a long time. Perhaps since before you began using a password manager and were being so much more cautious about your choice of password. (In other words, Leo, those accounts where your identity is still being verified with the password "monkey.") Shutting down any long unused accounts will meaningfully reduce your online target profile. It's worth doing.

And assuming that your password manager allows you to see a list of everything it's holding for you, you might just take a moment to scan through it to see whether you really need all that. We all tend to accumulate a growing collection of "one off" sites where keeping an account might serve no further purpose. Again, reducing your target profile makes sense. If nothing else, have your password manager display those passwords and make sure they have been turned into a long string of gibberish.

As these massive lists age, they automatically lose their value as people change their eMail and hopefully their passwords. But we can all help those lists age-out even faster, by deliberately obsoleting the data that they contain about who we were. us. We cannot change what the lists say about our past, but we can arrange to have them say nothing about who we are today.

~~~

Next week, unless something even more tantalizing comes up, we're going to examine a brilliantly clever supply-chain attack that was fortunately pulled-off by a good guy security researcher... because if it hadn't been, all hell would have broken loose.
~~~