SECURITY NOW!

Transcript of Episode #857

The Inept Panda

Description: This week we're going to take a look at our law enforcement and cyber defense recommendations regarding safe conduct while in Beijing for the 2022 Winter Olympic Games. We're going to take a look at a serious CVSS 9.9 vulnerability affecting Linux's use of SAMBA, and at some interesting details of so-called "Living Off the Land" exploitation of commonly present operating system utilities. We'll examine Microsoft's most recent approach to application packaging and installation triggered by their recent wholesale neutering of its primary application and feature. And we're also going to celebrate a welcome change in Microsoft policy that's been 20 years in the making. I'll share a brief pre-announcement of a new forthcoming GRC quickie freeware utility. Then we'll take a close look at "MY2022," the iOS and Android application which all attendees of the Beijing Olympics are required to install, carry, and use. Citizen Lab's reverse-engineering analysis will explain how this week's podcast got its name.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-857.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-857-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. Yes, I'm back. Thank you, Jason, for filling in last week. We've got lots to talk about. A vulnerability with Linux's SAMBA that everybody's going to want to fix. It's a 9.9 on the Richter scale. We'll talk about living off the land, a way of exploiting commonly present operating system utilities. There's quite a long list. You might be curious to find out what that is. And then Steve's going to take a look at the application required of all Olympians, MY2022. Turns out it's a nightmare of security flaws. All coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 857, recorded Tuesday, February 8th, 2022: The Inept Panda.

It's time for Security Now!, the show where we cover the latest security news with Mr. Steve Gibson. He explains it all to us. Hello, Steve.

Steve Gibson: You are back.

Leo: I am back.

Steve: From a week off, and Jason and I...

Leo: Well, it was really only two days off. It just happened to hit on Tuesday and Wednesday.

Steve: Oh.

Leo: But Jason, thank you for filling in, Jason Howell. I appreciate it.

Steve: Yup.

Leo: And I hear I missed some important stuff. It's why you've got to listen to every show.

Steve: Well, maybe. As I was saying before, you did, you were on vacation when Tom and I did the what has become a significant podcast to explain from just soup to nuts how the bitcoin blockchain works. And so that turned out to be an important one to have seen. This one, last week's episode with Jason, explaining how Google's Topics functions, we can wait to see if it ends up getting any traction. If not, well, it was an interesting podcast.

Leo: Yeah. We'd talked about FLoC, and that didn't end up being the system. So, yeah.

Steve: Exactly.

Leo: They're just throwing stuff against the wall at this point.

Steve: So, yeah. This one, I mean, this is really different. This looks like they're seeing all the back pressure which is mounting against tracking. And they're saying, okay, you know, we're going to have to get on the anti-tracking bandwagon, period. It's going to have to happen. So they've got a really nice proposal which demonstrates they were listening to everything everyone was saying about FLoC and taking it seriously. So anyway, we'll see.

Leo: But what is on the agenda? Because I am here, so now you can talk about the good stuff.

Steve: Ah. Glad to have you. You're right. This is Episode 857 for Patch Tuesday of February, that is, the 8th. So we'll be talking about its aftermath next week. This week we're going to take a look at law enforcement and cyber defense recommendations regarding safe conduct in Beijing for the 2022 Olympic Winter Games. We're going to take a look. And they had some interesting things to say. I mean, nothing shocking, but still. It's interesting, sort of like what's begun to emerge around the Olympics every year. We're going to take a look at a serious CVSS rated 9.9 vulnerability, affecting Linux's use of SAMBA. And at some interesting details of so-called "living off the land" exploitation of commonly present operating system utilities.

We'll examine Microsoft's most recent approach to application packaging. Well, which was triggered by their recent wholesale neutering of its primary feature. Whoops. And we're also going to celebrate a welcome change in Microsoft's policy, a significant policy which has been 20 years coming. I'm going to share a brief preannouncement of a new forthcoming GRC quickie freeware utility. I think everyone's going to get a kick out of it. I'm sure I'll be finished with it. I already released it for testing after less than a day. But it's something that I've become convinced that ought to be done. And then we're going to wrap up with the title of the podcast, "The Inept Panda," taking a close look at what's called "MY2022." That's the iOS and Android application which all attendees of the Beijing Olympics...

Leo: Oh, yeah.

Steve: ...are required to install, carry, and use. Citizen Lab reverse-engineered the app.

Leo: Oh, dear. Oh, dear.

Steve: And we will be looking then at how the podcast got its name today, "The Inept Panda."

Leo: Oh, boy. I can't wait. I can't wait. That's going to be very, very interesting. All right. Picture of the Week time.

Steve: So this is one that's a little difficult to describe. It's so wonderful that I would commend our listeners to grab the first page of the show notes just to take a look at it. So it came as is. I didn't add the headline or anything. But the headline, the title, is "1st Rule of Programming: If it works, don't touch it!" And what we're seeing here is like a drainpipe coming down off of the side of a building. And I'm a little curious about how this could have ever worked because it looks like there's an elbow missing which would have converted the pipe to sort of downward, well, sort of sideways pointing out to downward to finish pouring in through the rest of the pipe. But it doesn't look like it could have ever been quite that way.

So again, it's not clear what the history is here. But the point is that we have a stream of water flowing out of this drain and sort of following a ballistic curve toward the ground. Well, as a consequence of the way this drainpipe is broken, it's completely overshooting the opening that you'd like to have it going to, like completely. Yet it is where it would be striking the pipe on the ground, where this arcing flow of water would be striking the pipe, someone, the programmer in this case, or the programmer equivalent, has chipped open the pipe so that the stream of water is perfectly going into a jagged hole in the pipe, but it's accomplishing its mission. And...

Leo: It sure is.

Steve: It works.

Leo: It works. Don't disconnect it.

Steve: So as they say, if it works, don't touch it. First rule of programming. It's like, is it working? Yes. Okay, fine.

Leo: Yes, okay, don't touch it, anybody.

Steve: Okay. So last Tuesday, upstream by a few days of the beginning of Beijing's 2022 Olympic Games, the U.S. FBI warned that visitors to this year's Olympic Games being hosted, which are being hosted in Beijing, would be well advised to leave any fancy electronics at home. Grab an inexpensive burner, the FBI literally said, "Get a burner phone." I'm sure they know about burner phones because the bad guys use them a lot. In this case, we're not bad guys. We're just people who don't want to have our high-end smartphones totally taken over.

Anyway, the FBI says "well advised to leave any fancy electronics at home. Grab an inexpensive burner phone that could be broken into pieces once you've returned to your local airport." In other words, the advice about traveling to China is similar to the traditional advice we've often discussed for attending the annual Black Hat and Defcon conferences in Vegas, which can be distilled to "Be afraid. Be very afraid."

Leo: Wow.

Steve: So the FBI didn't mention any specific threats. But they didn't need to because it's become understood that malicious cyber - and I'm quoting them. "Malicious cyber actors could use a broad range of cyber activities to disrupt these events." There's a tremendous amount of malicious cyber activity occurring unfortunately now during the Olympics. Recall that the Tokyo Summer Olympics was a similar mess with athletes' and attendees' personal phones being targeted, the Games' TV broadcasts were disrupted, and the personal data of volunteers and ticket purchasers for the Tokyo Olympics were leaked online. The Olympics are high profile, and this makes everything happening there, unfortunately, a target.

After Tokyo's 2020 Summer Games, in a bit of a public relations puff piece, NTT Corporation, which was responsible for providing the cyber protection services for the Games, wrote. They said: "The total number of security events that were blocked during the Games, including unauthorized communications to the official website, was 450 million." What? Okay. NTT added that "None were successful" - none of these 450 million attacks were successful - "due to cybersecurity measures in place."

Okay, now, who knows what they're counting as, quote, a "security event." Maybe packets? Because 450, really, it's not possible that there were actually 450 million separate cyber intrusions or intrusion attempts or whatever. And we've seen this before, right, where a number is like ridiculously high because they're quoting any random packet that like was rejected by the firewall or something. So okay. On the other hand, no one doubts that the Games were and will be a site of intense cyber rivalry. And it's looking like we're in for that given the early warnings that have been seen.

The FBI said that during the those Summer 2020 Games: "While there were no major cyber disruptions, the most popular attack methods were malware, email spoofing, phishing, and the use of fake websites and streaming services designed to look like official Olympic service providers." And actually the fact that they're talking about fake websites we'll see by the end of the podcast is significant because unfortunately visitors won't be protected from that, although they'll think they are being. And the FBI reminded us that during the earlier 2018 Winter Olympics in South Korea, cyber actors associated with Russia were very active, and launched the Olympic Destroyer, as it was known, attack that interfered with the Games' Opening Ceremony. Those attacks were enabled through a spear-phishing campaign and also malicious mobile apps. As we'll see, the mobile app in this case I think is not malicious, just very poorly written.

Okay. So as for taking a throwaway burner phone and leaving your personal phones and PCs at home, the FBI warned of potential threats associated with mobile apps developed by untrusted vendors. And our CISA said that, quoted the FBI, said: "The FBI urges all athletes to keep their personal cell phone at home and use a temporary phone while attending the events. The download and use of applications, including those required to participate or stay in-country, could increase the opportunity for cyber actors to steal personal information, install tracking tools, malicious code, or malware."

And of course it's not only our phones; right? Anything that's wireless, especially, I mean, I would be very circumspect about plugging anything into anything, either. But Bluetooth and WiFi can be suspect. The first step is to take the threat seriously. It deserves to be taken seriously. In the normal world, like just normal life, the actual likelihood of any individual being targeted is probably vanishingly small, as we say. We don't want to overhype things because there are times when you really do need to pay attention. But we've seen, you and I have talked about it, Leo, those maps of "cell towers," in air quotes, in Las Vegas weeks before and comparing them to during the Black Hat convention. It would be funny to see how many new fake cell towers had suddenly appeared during Black Hat, if it weren't so frightening. And of course those fake towers will be happy to field your connection. They don't care who you are. You're an opportunity, whoever you are. And the same has been true during each of the recent Olympic Games. And it's sure to be so again this year, especially in China.

So this week's podcast is titled "The Inept Panda" because we're going to take a look at what Citizen Lab found when they deeply reverse engineered China's official and, interestingly, must-use app for the 2022 Winter Games. But again, that's far from the only worry during the Games.

Okay. So maybe this podcast will get to some people who may be saved from, you know, just by, again, raise your shields. Be really careful. Turn off any stuff you don't need. Turn off Bluetooth. Turn off WiFi if you don't really need it. I would say free Internet is something you want to be very careful about.

So we have a serious CVSS 9.9 remote code execution vulnerability in SAMBA, and it impacts at least Red Hat, SUSE, and Ubuntu Linuxes running SAMBA prior to v4.13.17. That's just been patched by its maintainers. So that one is safe, 4.13.17. Also patched were the other release flows 4.14.12 and 4.15.5. Everyone using SAMBA is being urged to update to one of those three. So, and you probably know if you are because it's not something that's running in Linux typically by default. As far as I know. I know that it's not running in FreeBSD.

Leo: I think if you have file shares you're using, though, it's likely that you've using SAMBA. Right?

Steve: Well, so would Linux assume that file shares would be sharing with Windows? Because of course that's SAMBA's origin.

Leo: I mean, there's other - there's CIFS, and there's Apple's system.

Steve: Right. Well, CIFS was just an early version of SAMBA.

Leo: SAMBA, right. Yeah, SAMBA is an open version of SMB, the LAN manager.

Steve: Correct. And in fact that's, yeah, exactly. SAMBA got its name from SMB, you know, add an A after the first S and another one at the end. So SMB turns into SAMBA. And of course SMB stands for Server Message Blocks, which is the file and printer sharing protocol originally designed by Microsoft back in, as you said, the LANMAN era. It was the original file and printer sharing protocol. And unfortunately, I mean, it's good that it's still with us. But that does mean that it's old and creaky and probably brings along a bunch of legacy baggage and probably an embodiment of legacy thinking. I mean, that's really what's in a lot of these older protocols is just the way they were doing things back then. I mean, this predated the Internet; right? This was coax. And when a network interface was, you know, the card was like $1,500 for, like, per workstation. So, I mean, it was a whole different world.

Today SMB is in its third generation. And it is what Windows still uses. I have it running on all of my Unix machines since, as you said, Leo, it is so convenient to be able to use SMB to attach a Unix file system or Linux to Windows File Explorer and browse around in the nice Windows GUI and edit configuration files in a convenient Windows editor, all as if it was a local drive. But as I said, SMB is an old and complex protocol with security added as an afterthought. So you will never find it or any other overly complex protocol exposed to the public Internet by any of my servers, and I have said many times in the past nobody wants to have file and printer sharing open on the Internet. Mistakes are just too easy to make.

And today we have another biggie. In this case, the exploitation of this critical vulnerability would allow attackers to gain remote code execution with root privileges on any Linux servers offering this, or systems. If you've got a SAMBA daemon running, then you're serving that protocol; right? So any Linux machines offering the protocol. The ability to execute remote code as a root user, as we know, means that an attacker would be able to read, modify, write, delete any files on the system, enumerate its users, install malware, you know, cryptominers, ransomware, and also pivot to gain access deeper into a corporate network.

So this bug is being tracked as CVE-2021-44142. So it got numbered last year. It's specifically an out-of-bounds heap read/write vulnerability appearing in the VFS, that's the virtual file system module, which is called "vfs_fruit." Or just Fruit, for short. And we'll see where it got its name in a second. The vulnerability was used and disclosed by a pair of researchers from STAR Labs during the Pwn2Own Austin 2021 competition, which we covered last year. After the event, researchers from Trend Micro's ZDI, who host the Pwn2Own, they took a look at it more closely and discovered additional variants of the vulnerability.

And our old friend Orange Tsai of DEVCORE - remember he's the one who started all of last year's Exchange Server debacle by showing Microsoft the first of what turned out to be a great many problems in Exchange Server's code. Anyway, he had - assuming it's a he - had independently reported this problem directly to the SAMBA maintainers. So it rates a 9.9, which we know is, you know, it's not a 10.0, but it's really close, because it's one of those no-authentication-needed exploits which can provide full root remote access if you happen to have this VFS Fruit module running.

Okay. So what about that? The Fruit module obtained its name due to the famous fruit-named company whose clients, yes, it was created to converse with. Yes, that's right, the Fruit module that ships with SAMBA is designed to provide interoperability between

SAMBA and Netatalk, with Netatalk being an open-source implementation of the AFP Apple Filing Protocol which is used to converse with macOS clients. So when everything's in place and working, it allows Unix-like systems (Apple's) to serve as file servers for Apple devices. Once a session is established, SMBD, which is the SMB daemon, allows an unauthenticated user to set extended file attributes.

And therein lies the problem. It is in the ability to write extended file attributes over the Apple Filing Protocol enabled by the VFS Fruit module that allow - there is a heap overflow which allows you to then perform a buffer overflow and leverage that into execution. So as always, the right solution is to update immediately to one of the releases that has been repaired. If the Fruit module is not being used, then there's no vulnerability. You don't have anything to worry about. If it's present, but not actively needed, it can be removed from the SMB configuration and the SAMBA daemon restarted in order to get rid of it. So you don't want it running.

If you have to have it, if you have to use it, first of all, I don't know why it would ever be exposed to the public Internet. But things happen. You definitely want to update SAMBA. There's no question this thing will be instrumented, weaponized, and people will start scanning for it immediately.

Living Off the Land. This is just my favorite term. Really. When bad guys gain some source of presence on a system, no matter how that may initially have happened, they then typically need some means of doing something, right, once they're there. Either they need to arrange to obtain the tools they will need from some remote hosting server, or they need to use and abuse what's already present where they are.

The term, which as I said I really love, that the security industry has coined for the latter is known as "living off the land," meaning use what you've got, wherever you are, cleverly reusing an environment's existing crop of utilities to get up to some mischief. The "living off the land" phrase has been shortened to LOL, and thus these already present - yeah, I know. And thus these already present binaries, when used in this fashion, are known as LOLBins, phrased and written as a single word.

The obvious advantage to reusing a system's existing LOLBins for nefarious purposes is that they're already there. They're trusted by the system. They probably have the rights that they need. And they're approved for use by whatever antimalware might be watching over the environment, trying to see what mischief is going on. So going out and grabbing something remotely incurs the risk of tripping an alarm when something new is pulled across the environment's network. And in tightly locked-down environments, application whitelisting might prevent an OS from running code that hasn't been signed with a valid digital certificate. In such environments, LOLBins that have already been approved for use are often able to, for example, open and run untrusted and unsigned utilities. They're trusted, so the presumption is anybody using them is trusted, and so they should have greater rights.

I recently encountered an analysis, assembled by the security firm Uptycs - which is spelled U-P-T-Y-C-S, Uptycs - of the most commonly used LOLBins, actually the top five, which they have seen employed to further subvert each of the top five for Windows, top five for Linux, and the top five for macOS. In Windows, the well-known regsvr32.exe and the rundll32.exe utilities have recently experienced spiking levels of abuse, with both being used extensively by the Qbot and the IcedID backdoor malware over the course of the last year. Similarly, bizarre as it might seem, the Loki and Agent Tesla spyware samples have been caught exploiting a vulnerability in Microsoft's Equation Editor, EQNEDT32.exe. And of course the more power Microsoft has added to PowerShell over time, the more ways bad guys are finding to abuse it.

So Uptycs' top five LOLBins for Windows, in order of descending popularity, are number one, top, is PowerShell because, boy, if you get a hold of that, you can do a lot of damage. Then we have MSHTA, Regsvr32, WMIC, and that equation editor. So now we have PowerShell being a nearly perfect tool for adversaries looking to compromise a system. It provides them with access to various Windows features which can be abused for downloading payloads, disabling Microsoft Defender and firewalls, executing fileless malware out of RAM and so on. So, yeah. You really want to keep PowerShell out of the bad guys' hands, if possible.

MSHTA is the Windows utility that executes Microsoft's HTML Applications with the file extension .hta or JavaScript and VBScript files. Adversaries are able to leverage mshta.exe for proxy execution, that is, execution on their behalf, of malicious .hta files, JavaScript, and VBScript. The infamous TrickBot malware, which we'll be hearing a lot about this hour, often used as a first-stage loader for ransomware and other payloads, has been leveraging mshta.exe for the past year. So it is in active exploitation.

Regsvr32 is in third place, a Windows built-in utility. Anyone who's done like serious work with Windows has probably been asked to use Regsvr32 to register a DLL with the system that then makes it available globally for other things to use. Anyway, it's a built-in utility that can be used to register and unregister service DLLs. Adversaries are able to abuse it to download scripts hosted on remote servers and execute it in memory. Both the Dridex and TrickBot malware families have used Reg32 to facilitate their infection routines.

WMIC is part of the Windows Management Instrumentation, the WMI system. WMIC is a command executive for WMI. Like PowerShell, it's quite powerful and has comprehensive features that provide a handy set of capabilities for accessing local or remote Windows system components. Once again, the Dridex malware leverages WMIC to execute rundll32 in the execution phase of its attack lifecycle, but adversaries may also abuse WMIC to - and I always want to say K-E-Y-M-O-U-S-E.

Leo: Me, too.

Steve: To achieve execution, discovery, and lateral movement inside of networks. And as I mentioned, that again, really odd, I think it was only 5% in their top five, but it made it into the top five, Microsoft's Equation Editor, EQNEDT32.exe. It turns out that it is being used by Agent Tesla and the Loki malware to execute their arbitrary code. So there's a flaw in there somewhere that they're able to leverage. And of course it's not just Windows that has plenty of LOLBins. Over on the Linux side, Uptycs' top five LOLBins are chattr, Wget, setfacl, crontab, and rm, which I got a kick out of.

This chattr function, as in Change Attribute, C-H-A-T-T-R, in Linux, is used to set and unset file attributes. Adversaries use this for changing the permissions of the file system files or to make their dropped files immutable to prevent users from deleting them. The self-propagating Kinsing malware uses this change attribute to change the permissions of SSH keys and password files in the defense-evasion phase of its attack lifecycle.

Linux's Wget function or command is so handy that I always have a Windows binary of it available for my own command line use. It's just too handy to be without. Unfortunately, the bad guys agree. They use it, as I do, as a no-nonsense means for quickly downloading files from across the Internet. Malware families like the Mirai botnet use Wget extensively to download the second stage of its malware.

Linux's setfacl, ACL as in Access Control List, is used, as you'd expect, to set, modify, or remove the access control lists which are used to control access to regular files and directories. Once again, the Kinsing malware that seems to be all about permissions, uses setfacl to set executable permission on bin/chmod in the defense-evasion phase of its attack lifecycle. And I'm wondering, I didn't dig into it any deeper, but chmod certainly already has its executable bit set because it's a command. So maybe the normal chmod, the real one, resides in a different executable directory, not underneath /bin. And so this thing is naming itself chmod and sticking itself under /bin and then using setfacl to turn on the executable permission bit in order to be runnable.

And of course the ever-handy "set it and forget it" crontab command easily opens the cron table for editing the list of tasks to be scheduled to run at specified times and intervals on the system. You know, it's very much like Windows Scheduler, sort of the same thing. Many a malware has arranged to come back from the dead through the clever manipulation of crontab's time-delayed command execution. In particular, cryptocurrency miners have been seen accessing cron entries to delete already-installed cron jobs, meaning get rid of the competition, and to install new cron jobs to keep themselves running.

And nothing says "erase your own footsteps" like "rm," Linux's short and sweet file removal command. Many malware families, including the Mirai and Gafgyt IoT botnets, as well as many cryptocurrency miners, depend upon rm to self-destruct and delete their tracks. And of course the most classic of all hacker tricks is to delete the log files which Linux and Unix systems are famous for creating. And the macOS is not without its handy tools being abused by malware. Uptycs lists the Mac's top five LOLBins as: openssl, curl, sqlite, killall, and funzip.

Being the original SSL and TLS development and testing platform, which we've talked about often, OpenSSL is literally the Swiss army knife of security and certificate management and manipulation. I had the occasion to use it just the other day on one of my FreeBSD Unix servers, the new one that I was setting up to host our GitLab instance. I needed to check that the new certificate chain I had installed was working correctly. I didn't feel comfortable placing GRC's wildcard certificate on a new and not-yet-trusted server. If that were to ever get loose, that certificate, it would allow someone, anyone, to spoof any GRC.com subdomain. Not good.

So instead I asked DigiCert to make a certificate that would only be valid for the domain "dev.grc.com." There's nothing else like OpenSSL for dumping and diagnosing secure connection setup. Unfortunately, like any powerful tool, it can be just as powerful when in the hands of someone malicious. On macOS, the Shlayer malware often leverages OpenSSL in conjunction with Base64, using both to encode and decode malware, also to encrypt malware, to hide it from detection.

We've talked about the clever abuse of the "curl" command several times in the past. Being a longstanding command-line tool used for transferring data using various network protocols, and that's really where it shines, curl is much like Wget, although Wget, being short for "web get," has more of a web orientation and is able to do things like follow redirection chains which is beyond curl. That said, curl is insanely more versatile with its protocol support. It can be used to obtain data from servers that are offering the DICT, FILE protocols, FTP, FTPS, of course for SSL or TLS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET, or TFTP. In other words, pretty much anything you can imagine you're able to use curl against.

And of interest to the bad guys, unfortunately, curl is designed to work without user interaction, so it's perfect for malicious scripting and remote unattended use. So curl remains the go-to command for many users and scripts. On macOS, it's also a favorite of the Bundlore malware which leverages curl to download payloads while it's busily setting up shop on a new machine.

SQLite, I was sort of surprised to see that. That was number three for macOS of the top LOLBins. Of course, SQLite is a transactional SQL database engine present in macOS and increasingly in other OSes as well. I've got it on a bunch of mine. For example, I guess I'm using Postgres on the new FreeBSD machine. But anyway, it's often used to create databases that can be transported across machines. The macOS, again, Bundlore malware uses SQLite to retrieve the history of downloaded files from the Internet in the exfiltration phase of its attack lifecycle. And actually they probably mean the history of exfiltrated, like uploaded files.

Fourth on the list for macOS is killall, a handy utility also found on many Unix and Unix-like systems. I use it. When newsreading clients connect to GRC's newsgroup server, in typical Unix style, a new instance of a single client server is forked for each connected client. So you end up with just a gazillion little processes, all running, each one talking to one persistent user of the newsgroup server. There have been many times when I have needed everyone to obtain an updated copy of some filter code which requires all instances of this client, typically hundreds of them because people tend to leave their newsreaders running, to be restarted and reloaded.

The only way to do that, short of rebooting the server, would be, well, and rebooting the server would be overkill, is to simply use Unix's killall command to terminate those hundreds of forked processes all at once. Naturally, this nice command can often be put to nefarious use and macOS's Shlayer malware uses killall to kill the running script's terminal window after its bash script activity has been completed in a little bit of a kamikaze maneuver.

And, finally, the Mac's "funzip" utility, fifth of the top five, is able to extract the contents of .zip and .gzip files directly to output from archives or other piped input. Shlayer also uses it with the head and tail commands to extract a malicious binary with a password. So living off the land, indeed. All of those very handy commands are right there on our systems, which saves the bad guys from needing to bring them with. And as I noted at the start, it's also far more stealthy to simply use what's already there.

Leo: I'm going to have you stop for a moment.

Steve: I know, Leo, you're familiar with all of those.

Leo: Yeah. Oh, yeah, and I have it all running all the time. By the way, I love your pronunciation. When you said "SQLite," I thought, wait a minute, because I've always said "SQL Lite." But then I found an interview with the creator of it, and he says it's pronounced like the mineral, ess-cue-ell-ite. So anyway.

Steve: Okay. So in one of those situations which just begs the question, "What could possibly go wrong?," Microsoft decided some time ago that it was old-fashioned for users to have to first download an installer package, then install it. Wouldn't it be so much better to just let users install a new program directly from any website just by clicking on a link? Like I said, "What could possibly go wrong?"

Leo: What could possibly, possibly go wrong.

Steve: And as if in answer to that question, Microsoft has now completely disabled that handy-dandy facility after some very popular malware was overheard to comment: "Hey!

This is great!" For the past three months the hard-to-get-rid-of Emotet gang, with their TrickBot - I told you we'd be hearing about Trickbot a lot - and the BazarLoader malware, has been using and abusing this new and very convenient protocol to deploy malware on users' systems.

Microsoft's page describing MSIX, which is what this is called, MSIX, explains that: "MSIX is a Windows app package format" - it's actually four package formats - "that provides a modern packaging experience" - because that's what you want in your packaging is a modern experience - "to all Windows apps. The MSIX package format preserves the functionality of existing app packages and/or install files in addition to enabling new, modern packaging" - we're talking about software - "packaging and deployment features to Win32, WPF, and Windows Forms apps.

"MSIX enables enterprises to stay current and ensure their applications are always up to date." Right, you know, because no one wants an old version of TrickBot. Anyway, "It allows IT Pros and developers to deliver a user-centric solution while still reducing the cost of ownership of applications by reducing the need to repackage." So, what?

Okay. To give us a better sense for what's going on here, I'll share Microsoft's three key feature bullet points. First, reliability. MSIX provides a reliable install, boasting, they say, a 99.96% success rate over millions of installs with a guaranteed uninstall. That's handy for malware. Network bandwidth optimization. MSIX decreases the impact to network bandwidth through downloading only - and this is what they say - the 64K block, as if we're supposed to know what that means, but okay. This is done by leveraging the AppxBlockMap.xml file contained in the MSIX app package. And they have a reference later to that in the article. MSIX, they said, is designed for modern systems and the cloud.

And the third point, disk space optimizations. With MSIX there is no duplication of files across apps, and Windows manages the shared files across apps. The apps are still independent of each other so updates will not impact other apps that share the file. A clean uninstall is guaranteed, even if the platform manages shared files across apps. And you know, if that sounds really familiar, it's because they tried this once before with something called DLLs. What a mess that turned out to be.

Leo: Oh, yeah. Still.

Steve: Oh, lord.

Leo: Turned out? It's still.

Steve: I know. Well, in fact that whole XSS, or SSX folder they have now, they call it the Windows Side by Side, SXS, it's huge. And basically what they did was they said, okay, that was a bad idea, those DLLs. So now we're just going to give every app its own set so that no one, like they just completely abandoned that whole idea. But I suppose their institutional memory has been lost, since all the folks who did that to us have since left Microsoft. So we're going to go through this all over again.

In any event, Microsoft apparently realized that their traditional app installing system was inherently introducing a great deal of bloat and redundancy. And of course now we have the cloud. So rather than downloading a big blob that the OS's app installer will then open, look around in, dig around through, read and deal with, let's make that remote. Let's create a new protocol that allows that bloated and redundancy-filled app install package to remain in the cloud on remote servers.

So now the use of a special scheme, you know, like HTTPS is a scheme, the use of a special scheme and file extension will establish an interactive session to the cloud that allows the installer running in the Windows client to first download only the installation manifest and then decide, based upon what it already has, only which additional components of the entire package it subsequently needs to ask for. Okay. Kind of sounds like, I mean, I can see where they went with this idea. We got the cloud. Let's not download a blob and only take a few pieces from it. Let's just get what the blob contains and then decide what more we want.

And Microsoft really put a lot of time and effort into this. This MSIX SDK is open source. The whole thing. The whole definition and project is open source. And although it was designed for Windows 10, it's not restricted to Windows 10. There's an MSIX Tech Community, and lots of additional resources. That initial AppxBlockMap.xml package is a document that contains a list of the app's files along with indexes and cryptographic hashes for each block of data that's stored in the package. The block map file itself is verified and secured with a digital signature when the package is signed. The block map file allows MSIX packages to be downloaded and validated incrementally, and also works to support differential updates to the app files after they're installed. I mean, this thing has everything.

Another file, AppxManifest.xml, is a package manifest document that contains the info the system needs to deploy, display, and update an MSIX app. This info includes package identity, package dependencies, required capabilities, visual elements, and extensibility points, whatever those are. Maybe you now get points for extensibility. I don't know.

There's also an entire file and registry - get this - a file and registry virtualization layer which provides a means for an application to declare that some set of its files and registry entries should be visible to other apps, and those should persist after app uninstall. Other files and registry entries are not visible to other apps and are removed on uninstall. Wow. Complicated much? I mean, is it any wonder this whole system is getting kind of brittle and flaky feeling? Anyway.

I mentioned there are four packaging formats. Those are reflected in the four file extensions: .msix, .msixbundle, .appx, and .appxbundle, some of which you may have seen. I've seen them sometimes in PowerShell stuff that I've had to do. So a huge amount of industry has been invested in this for Windows 10 and beyond. And Microsoft has just been forced to switch off the remote install, which was the entire point, due to its continued abuse.

In late November last year, the operators of the Emotet malware botnet started abusing the remote ms-appinstaller scheme - that's the scheme, ms-appinstaller - links in attacks targeting enterprise users by sending emails which lured innocent victims to specially crafted websites. These sites would claim to contain important documents that recipients needed to view, but for which they needed to install - wait for it - a PDF component which was missing. This was often made more convincing by arranging to continue a preexisting email dialogue.

And remember last year when all this was happening we were wondering how attackers might benefit from those multiple Exchange vulnerabilities which allowed them to obtain previous emails. Now we know. They would make spoofing attacks far more convincing by picking up a dialogue where it left off. Of course the dialogue that was left off was with the real individual at the other end of the email conversation, not the bad guys.

Anyway, the link provided for the PDF component would actually be an ms-appinstaller:// scheme that claimed to install an Adobe-signed file, but in reality it installed a version of TrickBot's BazarLoader malware, which is the beginning of the end for that system and probably everything it's hooked to.

Now of course this fancy new app packaging system has everything signed with cryptographic signatures. I mean, like out the wazoo. But that didn't stop the Emotet gang. Microsoft was finally forced to completely disable this protocol after all this work went into it because the gang found a way to spoof all those signatures in MSIX-packaged files. Those signatures appeared to be, to Windows, completely authentic. And I'll just note that it is unclear to me how anything like that could be spoof-proof. I mean, how are signatures being verified? If third parties are able to sign their own packages, and they must somehow be able to do that, then what prevents any third party from being malicious? Nothing.

Anyway, without having dug into this any more deeply, it appears that what happened is that Microsoft developed a technology that's too easy to use and abuse, and which cannot actually be usefully protected. Last year they delivered a patch for this problem, tracked as CVE-2021-43890, back in December as one of the 67 things they fixed at that time in December's Patch Tuesday. Except we now know that they didn't actually fix it, since attacks have continued to take place. And it would seem to me that it's actually not feasible for them to fix it.

So no more remote install, which means that all existing ms-appinstaller:// links have just died, world-wide and enterprise-wide everywhere. Microsoft wrote: "If you utilize the ms-appinstaller protocol on your website, we recommend that you update the link to your application, removing ms-appinstaller:// so that the MSIX package or appinstaller file will be downloaded instead onto its user's machine." Oh, just like in the old days. "We recognize that this feature is critical for many enterprise organizations. We're looking into introducing a Group Policy that would allow IT administrators to reenable the protocol and control usage of it within their organizations." In other words, IT admins may eventually be able to reenable this for local app deployment. But it sure looks like its use over the Internet is likely gone for good.

They've not said it's gone for good. A lot of the press coverage that didn't look at this as closely said, oh, yeah, it's been temporarily disabled. It's like, okay, well, let's see if it comes back. Because again, as I said, I can't see the model which allows this to be done safely. Apps are signed. Over time they are trusted by - those signatures gain trust and reputation by the various Windows Defender and other things that are looking at them.

But the goal here is a targeted attack. The idea with a signature on an app is that it's going to acquire a reputation. The publisher gets a reputation which allows the publisher's stuff to be trusted. Bad guys don't need that. They're able to do, you know, like a single attack makes it worth any amount of overhead. And it looks like this is one of the weaknesses of this system. So anyway, I expect we're not going to see it again. We'll see.

Soon Internet-sourced macros WILL NOT RUN - I had that in all caps - in Office apps. And then I had to double-check my spelling of "hallelujah." My god, Microsoft is slow to fix obvious problems which hurt their users. For how long has it been painfully obvious to everyone else that allowing macros to run in Office documents received from the Internet was a really bad idea? And I'll answer that question. There's never been a time when it wasn't painfully obvious. The inherent danger has always been clear.

Since the early 2000s, Microsoft has attempted to give unwitting users control over this by showing a mild and non-specific security warning in a toolbar at the top of the document. It stated that "Some active content has been disabled," alongside a button labeled "Enable Content." So, oh, it's been disabled? I guess I press the Enable Content button to turn it back on again. Yes. How many people do you imagine clicked the button in order to get what they believed they needed? What many got was a lot more than they bargained for.

Yesterday, in apparent reaction to a 20-years-delayed epiphany, Microsoft suddenly announced that, as of version 2203, starting with the Current Channel, the Preview, early this April, Access, Excel, PowerPoint, Visio, and Word would not - and I mean NOT, bold, caps - allow macro scripts to be enabled inside untrusted documents that had been downloaded over the Internet. That's huge. Not being able to enable, rather than merely being warned and told essentially to click "Enable," will make all the difference in thwarting spoofing attacks. Microsoft said that at a future date to be determined, they also plan to make this same change to Office LTSC (the long-term servicing channel), Office 2021, 2019, 2016, and 2013.

So whereas before the bar said "Security Warning" with a yellow exclamation point, the new bar says "SECURITY RISK" in all caps with a red "X" and the further explanation: "Microsoft has blocked macros from running because the source of this file is untrusted." And there ain't no "I trust it." There's a Learn More button which nobody wants to learn anything, so they're not going to push that. It's like, oh, okay. Well, I guess I'll see what this does without those macros, whatever they are.

Okay. So this will, without question, put a serious kink in the capabilities of malware gangs who've been relying upon tricking users into enabling the execution of macro scripts as a way of permitting those scripts to install malware on their systems. So as I said, hallelujah. It's really going to make a huge difference. And it's a bit odd to see Microsoft now confessing how bad it's always been.

In their announcement they wrote: "For years" - yes, years - "Microsoft Office has shipped powerful automation capabilities called 'active content.' The most common kind are macros. While we provided a notification bar to warn users about these macros, users could still decide to enable the macros by clicking a button. Bad actors send macros in Office files to end users who unknowingly enable them." You know, because there's a button there. Guess I should push it. "Malicious payloads are delivered, and the impact can be severe" - yeah - "including malware, compromised identity, data loss, and remote access."

So Microsoft, you're just figuring this out now. Okay. But I'm not going to look a gift horse in the mouth. Better late than never. The logic tree gauntlet that macros will now need to run finally gives them the respect their power should have always commanded. I mean, it's like this one, two, three, four, five, like a seven-stage "if" tree where you've got to take every branch correctly. In the start box it says "User opens file with VBA macros & MOTW attribute." Okay? What's MOTW? That stands for Mark of the Web, which is a flag Microsoft automatically tags files with when they've come from the Internet. And I'm sure our listeners will have seen those pop-up warnings when Windows is aware that a file they're about to execute came from the Internet. That's the Mark of the Web. That's the sign of the devil, the mark of the web.

Now it also turns out that most of us have always been victims of the "tyranny of the default" since, believe it or not, there's always been a Group Policy setting named "Block macros from running in Office files from the Internet," which enterprises have been able to turn on. But of course it's been off by default, despite the fact that Microsoft says that they recommend enabling this policy. But are they going to do it? No. Now they say that, if you do, your organization won't be affected by this upcoming change in Office's default behavior. Hopefully our IT admin listeners are already ahead of the curve. They turned that on. Nothing to see here.

The good news is across all of Office back to 2013, starting in April, this will be turned on. Yay. And again, how many times have we talked about Excel macros, PowerPoint macros, macros in any of this Office stuff, Word of course, doing bad stuff? And it's funny because the thing you - the document that you receive, the bad guys will have set it up so that the page you're reading knows macros are turned off. So it says, oh, in order to view the rest of this content, press that nice little button up there in the upper right. And then we'll be able to get going here. And of course people go, oh, and press it, and then they're in trouble.

So, and you seem fascinated by that chart, Leo. I also zoomed in and read it carefully because, I mean, it's like, "Is document from a trusted location?" Yes. Then okay. Macros enabled. No. Next. "Is macro digitally signed & trusted publisher on the PC?" Yes. Okay, fine. Macros enabled. No. Next. "Cloud policy to block?" And, I mean, and so on and so on and so on. So yes, this is just - it's wonderful that this kind of power is being managed the way it should have been because so many people have been hurt.

Okay. For many months, I guess unsurprisingly, I've been ignoring a continual stream of questions asking whether, and hopefully when, I would be offering Never11. And Leo, I still remember the first time you heard the name Never10.

Leo: Yeah.

Steve: You almost fell off your ball. I loved that you got a big kick out of that. Never10. Anyway, Never 10's become a bit famous. I think it has 3.5 million downloads, something like that.

Leo: Wow, wow.

Steve: And at the moment, none of my own systems qualify to move to Windows 11. Actually, it turns out I just found a laptop that I have that does. But we know that this whole qualifying for Windows 11 is a moving and totally arbitrary limitation. Microsoft allows virtual machines to install Windows 11 without any complaint over any processor and without any TPM. Windows 11 can run on anything. There's even a registry key, remember? The registry key which Microsoft created is named "AllowUpgradesWithUnsupportedTPMOrCPU." It's in there. So we're currently playing a game titled "We'd like to sell you some new hardware, so we're going to dangle Windows 11 in front of you and hope you bite." But it's Windows 11 that bites, at least as it's currently being offered.

I'm not being overly curmudgeonly about this. I listen to Paul and Mary Jo, both past Windows enthusiasts, every week scratching their heads and bemoaning in bewilderment what Microsoft is thinking with all of this. They just, it's like, it's just pretty funny to look at Paul just like, "Oh, Leo. I don't know." Anyway, now he's entertaining himself with Android and iOS and things because I think Windows is wearing a little thin.

Anyway, as for Windows 11, I think Microsoft is bound to eventually want all of us to move there. Lord knows they were willing to do battle to get everyone moved up to 10. So I predict that they'll eventually discover that Windows 11 is stable everywhere. What do you know? And they're going to want us all to move there. But I really don't want that. I don't want to go there until they put back a bunch of the features that they've taken away from Windows 10 which I like. So I realized that I wanted a Never11 app for my own use. And Lorrie certainly feels similarly. She doesn't want anything to change, either.

Looking around the 'Net last week, I saw a huge mess. There are Group Policy editor suggestions, which of course home users can't use because they don't have Group Policy editor. There's a wide variety of and many weird registry edit instructions, always accompanied by the obligatory cautions about the danger of editing the registry. And there are even sites instructing people to completely disable Windows Update, which we know is bad advice. It turns out that at the beginning of last September Microsoft published an optional update, KB5005101, which adds to Windows Update the explicit ability to tell Windows Update to target a specific edition of Windows and to remain there, and maybe to target a specific version of Windows that's still to be determined. Anyway, it is, of course, not enabled by default, but it's documented and honored.

So last Thursday evening I sent Mary Jo a note asking whether she knew whether the Enterprise editions of Windows would be subject to Microsoft's Windows 10-to-11 upgrade. Mary Jo replied the next morning that she assumed not, but that she would place a formal request with Microsoft to ask. And of course I was curious just to know whether it should be excluded or not.

So I worked on it last Friday. And by the end of the evening I posted a test release of Never11 for GRC's newsgroup gang to play with. Again, you know, I only did this because I knew it wasn't going to pull me away from SpinRite for long, and I had it done in less than a day. Since then, I've been refining my ideas for what I want it to be. I have an idea for something a bit more generic, with a different, yet still fun and memorable name. But I have a few more experiments to run first. I'm stealing time from SpinRite, which I want to be working on. And so this will be a quickie, but I think it's important enough, and will help enough people, that it's worth a couple of days. So I'll get it wrapped up and published shortly. And I should have something new, fun, and useful to announce next week.

I think the best way for me to begin will be to introduce Citizen Lab, the group who have carefully examined the smartphone app that everyone and I mean everyone attending the Beijing 2022 Winter Olympics is required to install and use. We've spoken of Citizen Lab many times in the past as their work has popped up in the security space from time to time. They're an interdisciplinary laboratory based at the Munk School of Global Affairs and Public Policy at the University of Toronto in Canada. Their focus is research, development, and high-level strategic policy and legal engagement at the intersection of information and communication technologies, human rights, and global security. So they seem to be well named.

They explain that they use a "mixed methods" approach to research combining practices from political science, law, computer science, and area studies with research which includes: investigating digital espionage against civil society; documenting Internet filtering and other technologies and practices that impact freedom of expression online; analyzing privacy, security, and information controls for popular applications; and examining transparency and accountability mechanisms relevant to the relationship between corporations and state agencies regarding personal data and other surveillance activities.

So, you know, not really the EFF, but sort of a Canadian take on responsible citizenship on the Internet. So they're perfect for this particular project, perfectly positioned to be interested in understanding the detailed operation of the smartphone app that all 2022 Olympic athletes and attendees and the press are being required to install and use.

So they begin their report with a bit of background to set the stage. Here's what they said. It's certainly useful. They said: "The 2022 Winter Olympic Games in Beijing have generated significant controversy. As early as February 2021, over 180 human rights groups had called for governments to boycott the Olympics, arguing that holding the Games in Beijing will legitimize a regime currently engaging in genocide against Uyghur people in China. Some governments, including Canada, the United Kingdom, and the United States, have pledged to diplomatically boycott the Games, meaning that these countries will allow athletes to compete at the Games, but will not send government delegates to attend the event.

"The International Olympic Committee (IOC), the organization responsible for organizing the Games, has been criticized for failing to uphold human rights. In December 2021, the United States House of Representatives voted unanimously to condemn the IOC and stated that the IOC had violated," you know, and blah blah blah. So lots of political controversy associated, unfortunately, with something that could use a lot less of that.

Skipping down, they explain: "Internet platforms operating in China are legally required to control content communicated over their platforms or face penalties. Vague definitions of prohibited content are often called 'pocket crimes,' referring to authorities being able to deem any action as an offense. Such crimes are utilized by the Chinese government to restrict political and religious expression over the Internet. Chat and other real-time communications platforms operating in China typically perform automated censorship using a block list of keywords" - and we'll see why that is important in a minute - "keywords whose presence in a message will trigger its censorship. Previous work has found little consistency in what content different Chinese Internet platforms censor. However, Internet platforms are known to receive censorship directives from various government offices or officials.

"In this report we analyze MY2022, an app required to be installed by all attendees to the 2022 Olympic Games, including audience members, members of the press, and competing athletes. The app is multi-purpose, implementing a wide range of functionality including real-time chat, voice audio chat, file transfers, as well as news and weather updates about the Olympic Games. The app can also be used to submit required health customs information for those visiting China from abroad, which includes submitting passport details, demographic information, as well as travel and medical histories." So I have to say, as I was plowing through this and being refreshed, it just sort of seemed to me unfortunate that the Olympics are being held in Beijing, but that's where they are.

Okay. So according to the Chinese government's official guide on the Games, MY2022 was built by the Beijing Organizing Committee for the 2022 Olympics. Public records and app store information show that the app is owned by a state-owned company called Beijing Financial Holdings Group. This app, MY2022, has a wide range of functionalities including tourism recommendations, GPS navigation, and COVID-19-related health monitoring. One of the functions MY2022 includes is to collect a list of medical information for health monitoring, which includes users' daily self-report health status, COVID-19 vaccination status, and COVID-19 lab test results.

Now, I was sure that the app would be supported on Android, and I assumed it would also need to be on iPhone. So I pulled the app up in Apple's App Store. The top two hits when searching for MY2022 were the app and another generically titled "Olympics." And I put both screen shots in the show notes. The official Chinese "MY2022" had 14 reviews and averaged two stars, whereas the generic "Olympics" app had 39,000 reviews averaging just a tad shy of a full five stars. Which if anyone's ever thought about that, requires that everybody pretty much give it a four- or a five-star not to pull it way lower. It looks like about 4.75 out of five. Okay. So clearly, MY2022, though apparently you must have it and use it, it isn't really what people are getting excited about.

Leo: I suspect the reviews will flood in after the end of the Games. Right? You're not going to review it poorly now. You're still in China.

Steve: That's a good point. I'm just not, you're right, Leo, I'm not used to Big Brother looking over my shoulder.

Leo: Yeah.

Steve: It really is creepy.

Leo: In fact, I would put a review in that says this as "This is fabulous. Five stars. Love you CCP. Keep up the great work."

Steve: Best thing I've used this year.

Leo: Yeah.

Steve: Yeah. Okay. So what do we know about the app that everyone is carrying around in their pockets for these next two weeks? Here's Citizen Lab's four-point summary, just to condense it. They said, first point: "MY2022, an app mandated for use by all attendees of the 2022 Olympic Games in Beijing, has a simple but devastating flaw where..."

Leo: Oh, boy.

Steve: I know. Let's just start right off with the good news, "a simple but devastating flaw." And remember what this thing is doing; right? I mean, it's like all of your health information, your COVID-19 status. You have to check in with it every day and blah blah blah. And chats and text and voice chats. Anyway, "...devastating flaw where encryption protecting users' voice audio and file transfers can be trivially sidestepped. Health customs forms which transmit passport details, demographic information, and medical and travel history are also vulnerable. Server responses can also be spoofed, allowing an attacker to display fake instructions to users." What could possibly - yeah, anyway.

Point two: "MY2022 is fairly straightforward about the types of data it collects from users in its public-facing documents. However, as the app collects a range of highly sensitive medical information, it's unclear with whom or which organizations it shares this information."

Leo: Well, that's the beauty of the CCP. You don't need to ask.

Steve: That's right.

Leo: They're going to take care of that for you.

Steve: That's right. Hey, it's our country. You're a visitor. You play by our rules.

Leo: They may, to be fair, have different standards for privacy. In fact, I'm willing to bet they do in China. So, you know, we're the crazy privacy people.

Steve: That's true, although we'll see here that this thing even violates China's privacy guidelines.

Leo: Oh, that's funny.

Steve: So "MY2022 includes features that allow users to report 'politically sensitive' content. The app also includes a censorship keyword list, which, while presently inactive, targets a variety of political topics including sensitive domestic issues as well as references to Chinese government agencies.

"While the vendor did not respond" - here we're getting into the meat now - "the vendor did not respond to our security disclosure, we find that the app's security deficits may not only violate Google's Unwanted Software Policy and Apple's App Store guidelines, but also China's own laws and national standards pertaining to privacy protection, providing potential avenues for future redress."

Okay. So "For domestic Chinese users, MY2022 collects personal information including name, national identification number, phone number, email address, profile picture, and employment information, sharing it with the Beijing Organizing Committee for the 2022 Olympics. For international users, the app collects a different set of personally identifiable information including users' demographic information and passport information, issue and expiration dates, as well as the organization to which the individual belongs.

"The official Olympic Games Playbook introduces MY2022 as a smartphone application for, among other things, health monitoring. MY2022 outlines in its privacy policy that it collects and uses users' daily self-report health status" - oh, I'm feeling just fine, thanks, no problems here - "COVID-19 vaccination status, and COVID-19 lab test results for such purposes. While the official Olympic Games Playbook outlines that personal data such as biographical information and health-related data may be processed by a list of entities including the Beijing 2022 Organizing Committee, Chinese authorities (including the Chinese National Government, local authorities, and other authorities in charge of health and safety protocols), the International Olympic Committee, the International Paralympic Committee, and 'others involved in the implementation of the COVID-19 countermeasures'" - okay, so anybody who wants it - "MY2022's privacy policy itself did not specify with whom or with which organizations it would share users' medical and health-related information." So could be anyone.

"Similar to other China-based apps Citizen Lab had studied in the past, MY2022 outlines several scenarios where it will disclose personal information without user consent, which include but are not limited to national security matters" - and we know how broadly defined those can be - "public health incidents, and criminal investigations. MY2022's privacy policy did not specify whether each disclosure would be conducted under a court order, or which organizations would potentially receive information."

Now, here's what's bizarre. Citizen Lab examined version 2.0.0 of the iOS version of MY2022 and version 2.0.1 of the Android version of MY2022. As we know, verifying the authenticity of security certificates is one of the fundamental requirements of secure end-to-end encryption. In fact, without verifying the identity claim being made by whomever you are connecting to, a man in the middle or anyone spoofing the server at the other end of the connection will be able to decrypt and see everything in plaintext. Without authentication, encryption is truly meaningless. Verifying certificate authenticity is so crucial that both the iOS and Android platforms of course build this into their APIs. In fact, it's difficult to imagine this not being done.

Yet despite this, Citizen Lab discovered that many of the certificates protecting the connections to the app's critical backend services and servers were not being checked for authenticity. Although we don't know exactly what each server does, the names of those certificates which are not being checked seem quite important. We had my2022.beijing2022.cn. Also tmail.beijing2022.cn. Also dongaoserver.beijing2022.cn, app.bcia.com.cn, and health.customsapp.com. None of those being checked.

The connections to those servers are using certificates. They are TLS-encrypted. But the authenticity of the certificate being provided to the application by the remote server is never checked. It's difficult to imagine how this could be anything other than incredibly sloppy coding. And again, that's why I just said the "Inept Panda" rather than the malicious or the evil or the sneaky or something. I mean, it just, like, what? And the biggest problem is that, as I noted at the start of this podcast, the International Olympic Games have become attacker-central. And this failing to authenticate has been all over the news since Citizen Lab reported this publicly exactly three weeks ago today. So it's not as if the bad guys, A, don't care; or, B, don't know. They both care and know.

Adding support for the idea that this was just incredibly sloppy coding rather than deliberate subterfuge is the additional observation that some sensitive data was also being transmitted without any SSL encryption or any security of any kind at all. Citizen Lab found that the MY2022 app transmits non-encrypted data to "tmail.beijing2022.cn" on port 8099. These transmissions contain sensitive metadata relating to messages, including the names of messages' senders and receivers, and their user account identifiers. Such data can be read by any passive eavesdropper, such as someone in range of an unsecured WiFi access point, someone operating a WiFi hotspot, an Internet Service Provider, or other telecommunications company. It's in the clear, in plaintext.

Now, being responsible, and just hoping to be able to get these important oversights fixed before the Games began, Citizen Lab privately disclosed their findings, being responsible, to the Beijing Organizing Committee for the 2022 Olympic and Paralympic Winter Games back at the beginning of December, on the 3rd, of 2021. December 3rd. The disclosure indicated that there would be a deadline of 15 days to substantively respond to the disclosure and 45 days to fix the issues. Because the clock was ticking; right? The Games are now.

And let's remember that none of this is rocket science. It had to have simply been an oversight. So it would presumably take whomever had written that code and somehow failed to verify those servers' certificate authenticities, you know, a lazy afternoon to add that to the existing codebase.

But after waiting a month and a half, as of three weeks ago on January 18th, Citizen Lab had received no response of any kind to their disclosure. So as they said they would, they went public with their findings. And the result of that was the quite predictable massive kerfuffle when all of this was picked up by the tech and other secondary press.

The day before Citizen Lab's deadline-driven disclosure, on January 17th, update v2.0.5 of the iOS version of MY2022 appeared in the Apple App Store. Thinking that it might have been the awaited update which fixed the problems, and that the app's authors were just not much for chitchat, Citizen Lab promptly examined the update application.

Not only have none of the known and documented and previously reported problems been resolved, but the app had introduced a new feature called "Green Health Code" whose data transmissions were also vulnerable to interception and spoofing because, although encrypted, they also failed to verify the authenticity of the server's certificate.

The "Green Health Code" feature asks for travel document information and medical history information similar to the information they had already found to be insecurely transmitted by the app's vulnerable customs health declaration feature.

And on the censorship side, according to MY2022's description in Apple's App Store, the app implements a wide range of communication functionalities including, as we said, real-time chat, newsfeeds, and file transfers. In previous studies, Citizen Lab found the presence of censorship and surveillance keyword lists in different Chinese communication apps that provide similar services. Bundled with the Android version of MY2022, they discovered a file named "illegalwords.txt" which contains a list of 2,442 keywords generally considered politically sensitive in China.

However, despite its inclusion in the app's package, they were unable to locate any functionality where those keywords were actually used to perform censorship. The people who built this app seem to be so clueless that they may have simply forgotten to engage the word filtering function, if indeed there was any.

Leo: We built it in, but we didn't turn it on.

Steve: Yeah. So it's like, oh, we're sorry. You know, it's in there; but oops.

Leo: Whoops. That's hysterical.

Steve: Unbelievable. So it's unclear whether this keyword list is entirely inactive; and, if so, whether the list is inactive intentionally. However, the app contains code functions designed to apply this list toward censorship, although at present those functions do not appear to ever be called anywhere. So who knows? We'll never know.

A spokesman for the International Olympic Committee justified the app's security issues - Leo, he explained this apparently, however, without understanding them at all, by saying that due to the COVID-19 pandemic - where did that originate? Oh, never mind. Due to the COVID-19 pandemic, "special measures," that's what it said, "special measures" needed to be put in place. That's it. What? This individual also defended the app...

Leo: Yeah, he misunderstood the criticism, obviously, yeah.

Steve: Yeah, by saying it received approval from the Google Play store and the App Store.

Leo: Well, that's a point, good point.

Steve: Well, okay, then, fine. And adding insult to injury, as if they hadn't already been clueless enough, the IOC said: "A closed-loop management system has been implemented." This is the IOC, the International Olympic Committee. "A closed-loop management system has been implemented. The MY2022 app supports the function for health monitoring. It is designed to keep Games-related personnel safe within the closed-loop environment."

Leo: Yeah. They're responding to the wrong question.

Steve: Yeah. So I cannot imagine having to install such an app on my phone in order to attend our world's Olympic Games anywhere.

Leo: What's the risk? What's the chief risk, do you think?

Steve: Okay. So with an app - okay. So Apple's very good about containment, as we know. I mean, they go to great lengths to contain an app. On the other hand, we know that they're not perfect. Android has more problems with cross-app contamination. And there are things like hooking the keyboard, hooking the camera, turning the device into spyware. We know, for example, that Pegasus has no problem doing that time after time after time on iOS or Android.

So I just, you know, the advice would be what we started off talking about, what the FBI recommended, leave your Apple and Android stuff. Leave your fancy phones home. Go to the drugstore and grab off a J-hook a $40 burner phone and just take that so that you can take pictures and create memories and text your family and so forth. Just don't bring, I mean, it's very much like the same advice about being super careful when you're in Las Vegas during Black Hat. I wouldn't bring my laptop. Just like, no, no, I'll just bring a Chromebook and do the expunge button or whatever it's called on a Chromebook.

Leo: And don't do anything private on that phone because it's all potentially stolen in transit.

Steve: Yes. And, you know, stick it in an RF bag maybe, you know, and don't let the camera see anything sensitive. I mean, really, just consider that your hotel room is bugged, and your devices are bugged.

Leo: It's a spy device, really.

Steve: Yeah.

Leo: But the worst thing is it's not a spy device just for the Chinese Communist Party. It's a spy device for anybody who wants to use it.

Steve: Yes. And the Olympics have proven to be a target-rich environment. I mean, there is a lot of attention being put into hacking people there. It's like a more global version of Black Hat.

Leo: I'm going to guess that the athletes there have other things on their mind, and they're probably not considering OPSEC at this point. One would hope that the people who manage the teams and who are taking care of these people would say, look, don't, don't.

Steve: Are saying do not, you know, you don't want to embarrass your country. So leave your phone in the hotel room.

Leo: Right. Steve, you've done it again. Another great week. I'm sorry I missed last week. But, hey, the good news is you can't miss a week because we record them all, and you can listen to them at any time. Steve is at GRC.com. That's where you'll find SpinRite, his bread and butter, the world's best mass storage maintenance and recovery utility. Also lots of free stuff like, soon, Never11. I mean, what is the timeframe for Never11, you think?

Steve: Oh, it'll be like in the next day or two.

Leo: It's not hard; right? It's pretty easy.

Steve: No. I've already published the first release, and it's up and running.

Leo: Okay. So it's there. All right.

Steve: Yeah.

Leo: ShieldsUP!, all that other great stuff. You can leave him a message there at GRC.com/feedback. And of course you'll find 16Kb audio versions of this show for the bandwidth-impaired, 64Kb for those with two ears. There also is a copy of a human-written transcription. Takes a couple days to get up there, but that's very handy if you like to read along as you listen, or to find stuff, to do searches. It's all there, GRC.com.

We have the show on our website, as well, audio and video, at TWiT.tv/sn. Of course there's a YouTube channel dedicated to the video. You can watch it there at any time. Or that's a good way to share it, too, with other people, if you see a thing that you want other people to know about. Maybe you've got a friend in the Olympics, you might want to just take that part of it and share it. YouTube has that show.

And then there's the podcast route. We make a podcast out of it so you can always subscribe in your favorite podcast player. You'll get it automatically each week, which is nice. Please, if your podcast player allows reviews, leave us a five-star review because we want everyone to listen to Security Now!. This is a must-listen, not just for security professionals, but for anybody interested in security and privacy online. We will be back next Tuesday, 1:30 Pacific, 4:30 Eastern, 21:30 UTC. You can watch us do it live, live.twit.tv. You can chat live at irc.twit.tv. Steve, have a great week. We'll see you next time.

Steve: My friend, see you for 858 next week.

Leo: 858. Wow. Bye-bye.

Steve: We're getting there. Bye.