Security Now! #857 - 02-08-22

The Inept Panda

This week on Security Now!

This week we're going to take a look at 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 which was triggered by their recent wholesale neutering of its primary feature. And we're also going to celebrate a welcome change in Microsoft policy that's been 20 years coming. I'm going to 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.

Security News

China's Olympics: Leave your tech at home

Last Tuesday, the US FBI warned that visitors to this year's Olympic Games being hosted in Beijing, China would be well advised to leave any fancy electronics at home. Grab an inexpensive "burner" phone that can 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 for attending the annual BlackHat and DefCon conferences in Las Vegas: Be afraid. Be very afraid.

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

After Tokyo's 2020 Summer Games, in a bit of a public relations puff piece, NTT Corporation, which provided cyber protection services for the Games, wrote: "The total number of security events that were blocked during the Games, including unauthorized communications to the official website, was 450 million" and NTT added that "none were successful due to cybersecurity measures in place." Now, who knows what they're counting as a <quote> "security event" — maybe packets? — because it's not possible that there were actually 450 million separate cyber intrusions. But no one doubts that the Games were and will be a site of intense cyber rivalry.

The FBI said that during the Summer 2020 Games: "While there were no major cyber disruptions, the most popular attack methods used were malware, email spoofing, phishing and the use of fake websites and streaming services designed to look like official Olympic service providers. And the FBI reminded us that during the 2018 Winter Olympics in South Korea, cyber actors associated with Russia were very active, and launched the Olympic Destroyer attack that interfered with the Games' Opening Ceremony. Those attacks were enabled through spear-phishing campaigns and malicious mobile apps.

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. The CISA said that "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. Anything with Bluetooth or WiFi should be suspect. The first step is to take the threat seriously. It deserves to be taken seriously. In the normal world, the real likelihood of any individual being targeted is probably vanishingly small. But we've seen the maps of "cell towers" (in quotes) in Las Vegas weeks before and then during the BlackHat convention. It would be funny to see how many new fake cell towers had suddenly appeared if it weren't so frightening. And those fake towers will be happy to field your connection. They don't care who you are. You are 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.

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 must-use app for the 2022 Winter Games. But that's far from the only worry during the Games.

We have a serious CVS 9.9 remote code execution vulnerability in SAMBA, which impacts at least Red Hat, SUSE and Ubuntu Linuxes running Samba prior to v4.13.17, which has just been patched by its maintainers. Also patched were versions 4.14.12 and 4.15.5. Everyone using Samba is urged to update to one of those three.

“Samba” gets its name from “SMB” by adding one 'A' after the ‘S’ and another at the end. SMB is the Server Message Blocks file and printer sharing protocol originally designed by Microsoft back in the early LAN Manager era. It was the original file and printer sharing protocol... and that means it’s old, creaky, and probably brings along some legacy baggage and an embodiment of legacy thinking. And its 3rd generation is that Windows still uses to this day, and I have it running on all of my UNIX machines since it's so convenient to be able to use SMB to attach a UNIX file system 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'll never find it, or any other overly complex protocol, exposed to the public Internet by any of my servers. Mistakes are just too easy to make... and here 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 protocol. The ability to execute remote code as a root user, of course means that an attacker would be able to read, modify or delete any files on the system, enumerate users, install malware — you know, cryptominers or ransomware — and also pivot to gain access deeper into a corporate network.

This bug (CVE-2021-44142) is specifically an out-of-bounds heap read/write vulnerability appearing in the VFS (virtual file system) module called “vfs_fruit.” The vulnerability was used and disclosed by a pair of researchers from STAR Labs during the Pwn2Own Austin 2021 competition. After the event, researchers from Trend Micro's ZDI, Zero-Day Initiative, discovered additional variants of the vulnerability, and our old friend Orange Tsai of DEVCORE (remember, he’s the one who started last year's Exchange Server debacle by showing Microsoft the first of a great many problems in that server’s code). Orange Tsai had independently reported this directly to the Samba maintainers. This rates a CVSS of 9.9 because it's one one of those no authentication needed exploits.

Interestingly, Samba's "Fruit" module obtained its name due to the famous fruit-named company whose clients 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. When everything's in place and working, it allows Unix-like systems to serve as file servers for Apple devices. Once a session is established, smbd — the SMB daemon — allows an unauthenticated user to set extended file attributes... and therein lies the problem.

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. If it’s present but

not actively needed, it can be removed from the SMB configuration and the Samba daemon restarted.

Living off the Land

When bad guys gain some sort of presence on a system, no matter how that may initially happen, they then typically need some means for doing something once they're there. Either they need to arrange to obtain the tools they need from some remote hosting server, or they need to use — and abuse — what's already present where they are. The term, which I really love, that the security industry has coined for the latter is known as "living off the land"... meaning to cleverly reuse 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 binaries, when used in this fashion, are known as LOLbins, phrased as a single word.

The obvious advantage to reusing a system's existing LOLbins for nefarious purposes is that they're already there, trusted by the system and approved for use by whatever anti-malware that might be watching over the environment. 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 white-listing might prevent an OS from running code that hasn't been signed with a valid digital certificate. But in such environments, LOLBins that have already been approved for use are often able to open and run untrusted and unsigned utilities.

I recently encountered an analysis, assembled by the security firm Uptycs (spelled u p t y c s), of the most commonly used LOLbins which they have seen employed to further subvert Windows, Linux and macOS systems. In Windows, the well known regsvr32.exe and rundll32.exe utilities have recently experienced spiking levels of abuse, with both being used extensively by the Qbot and IcedID backdoor malware over the course of the last year. Similarly, bizarre as it might first seem, the Loki and Agent Tesla spyware samples have been caught exploiting a vulnerability in Microsoft's Equation Editor (EE) EQNEDT32.exe. And, of course, the more power Microsoft has added to PowerShell, the more ways bad guys are finding to abuse it.

Uptycs' top five LOLbins for windows, in order of descending popularity are PowerShell, MSHTA, RegSvr32, WMIC and that equation editor, EQNEDT32.

PowerShell is 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 and so on.

MSHTA.exe is a Windows utility that executes Microsoft HTML Applications with the file extension (HTA) files or JavaScript/VBScript files. Adversaries are able to leverage mshta.exe for proxy execution of malicious HTA files, JavaScript or VBScript. The infamous TrickBot malware, often used as a first-stage loader for ransomware and other payloads, has been leveraging mshta.exe for the past year.

Regsvr32 is a Windows 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 regsvr32.exe to facilitate their infection routines.

WMIC.exe is part of the Windows Management Instrumentation, WMI system. Its quite powerful and comprehensive features provides a handy set of capabilities for accessing local or remote Windows system components. The Dridex malware leverages WMIC to execute rundll32 in the Execution phase of its attack lifecycle, but adversaries may also abuse wmic.exe to achieve execution, discovery or lateral movement.

And as I mentioned, that EQNEDT32.exe, Microsoft's Equation Editor helps the Agent Tesla and Loki malware to execute their arbitrary code.

And it's not just Windows that has plenty of handy LOLbins. Over on the Linux side, Uptycs top five LOLbins are: chattr, wget, setfacl, crontab and rm.

The chattr function in Linux is used to set/unset file attributes. Adversaries use this for changing the permissions of the system files or to make their dropped files immutable to prevent user deletion. The self-propagating Kinsing malware uses chattr to change the permissions of SSH keys and password files in the defense-evasion phase of its attack lifecycle.

Linux's WGET command is so handy that I always have a Windows binary of it available to my own Windows command lines. 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 utility is used to set, modify or remove the access control lists (ACLs) which are used to control access to regular files and directories. The Kinsing malware uses setfacl to set executable permission on /bin/chmod in the defense-evasion phase of its attack lifecycle.

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. 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 in miners have been seen accessing cron entries to delete already-installed cron jobs and to install a new cron job 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. The most classic of all hacker tricks is to delete the log files which Unix and Linux 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, 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, it would allow someone 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 to hide it from detection.

We've talked about the clever abuse of the "curl" command several times before. Being a long-standing command-line tool used for transferring data using various network protocols it's much like wget, though 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 for DICT, FILE, FTP, FTPS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET or TFTP. And, of interest to the bad guys, curl is designed to work without user interaction so it's perfect for malicious scripting. 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 is a transactional SQL database engine present in macOS and increasingly in other OSes as well. It's often used to create databases that can be transported across machines. The macOS Bundlore malware uses SQLite to retrieve the history of downloaded files from the Internet in the exfiltration phase of its attack lifecycle.

"killall" is a handy utility found on many Unix and Unix-like systems. When newsreading clients connect to GRC's newsgroup server, in typical Unix style, a new instance of a client server is forked for each connected client. There have been many times when I need everyone to obtain an updated copy of some filter code, which requires all instances of this client — typically many hundreds since people tend to leave their newsreaders running — to be restarted and reloaded. The only way to do that, short of rebooting the server which would be overkill, is to simply use Unix's "killall" command to terminate those hundreds of forked processes at once. Natrually, this nice command can often be put to nefarious use and macOS Shlayer malware uses killall to kill the running script's terminal window after its bash script activity has completed.

And the Mac's "funzip" utility is able to extract the contents of .ZIP or .GZIP files directly to output from archives or other piped input. Shlayer also uses it with head or tail commands to extract a malicious binary with a password.

So...

Living off the land, indeed. All of those oh-so-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.

The suspension of the ms-appinstaller:// protocol scheme handler
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?" 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 and BazaarLoader malware, has been using and abusing this new and so-convenient protocol to deploy malware on user systems. Microsoft's page describing MSIX explains that:

"MSIX is a Windows app package format [it's actually 4 package formats] *that provides a modern packaging 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 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!] 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."*

To give us a better sense for what's going on here, I'll share Microsoft's three key feature bullet points:

- *Reliability*. MSIX provides a reliable install boasting a 99.96% success rate over millions of installs with a guaranteed uninstall.

- *Network bandwidth optimization*. MSIX decreases the impact to network bandwidth through downloading only the 64k block. This is done by leveraging the AppxBlockMap.xml file contained in the MSIX app package (and they have a reference to later in the article.) MSIX is designed for modern systems and the cloud.

- *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.

You know, if that sounds really familiar it's because they tried this once before with something called DLL's. What a mess that turned out to be. But I suppose that their institutional memory has been lost, since all the folks who did that to us, have since left Microsoft. So I guess 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, now we have the cloud. So rather than downloading a big blog that the OS's app installer will then open, look around in, 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 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.

The MSIX SDK 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 is 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.

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.)

There's also an entire 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 that those should persist after app uninstall. All other files and Registry entries are not visible to other apps; and are removed on uninstall. Wow, complicated, much??

I mentioned that 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. 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 remote ms-appinstaller scheme 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. This was often made more convincing by arranging to continue a preexisting eMail dialog.

Remember that we were wondering last year how attackers might benefit from those multiple Exchange vulnerabilities which allowed attackers to obtain previous eMails? Now we know. They would make spoofing attacks far more convincing by picking up a dialog where it left off.

The link provided for the PDF component was actually an "ms-appinstaller://" scheme that claimed to install an Adobe-signed file, but in reality, it installed a version of TrickBot's BazaarLoader malware.

Now, of course, this fancy new app packaging system has everything signed with cryptographic signatures out the wazoo. But that didn't stop the Emotet gang. Microsoft was finally forced to completely disable this protocol because the gang found a way to spoof all those signatures in MSIX-packaged files. Those signatures appeared to be completely authentic to Windows.

And I'll note that it's 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 3rd party from being malicious? Nothing.

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. Except we know that they didn't actually fix it since attacks have continued to take place and it would seem to me that it's not fixable.

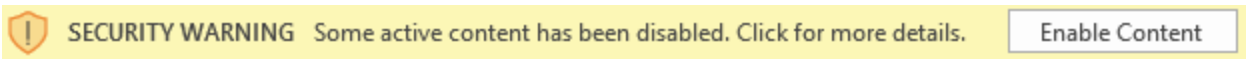
So, no more remote install which means that all existing ms-appinstaller:// links have just died. Microsoft wrote: *"If you utilize the ms-appinstaller protocol on your website, we recommend that you update the link to your application, removing 'ms-appinstaller:' so that the MSIX package or App Installer file will be downloaded instead onto its user's machine. 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 re-enable the protocol and control usage of it within their organizations."*

In other words, IT admin may eventually be able to re-enable this for local app deployment. But its use over the Internet is likely gone for good.

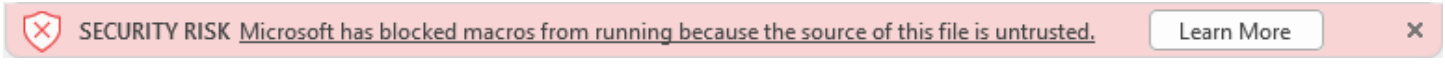
Soon: Internet-sourced macros WILL NOT RUN in Office apps!

And I had to double-check my spelling of "Hallelujah!!" My God Microsoft is slow to fix obvious problems which hurts their users. For how long has it been painfully obvious 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" 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 had bargained for.



Yesterday, in apparent reaction to a 20-years delayed epiphany, Microsoft suddenly announced that as of version 2203, starting with Current Channel (Preview) early this April, Access, Excel, PowerPoint, Visio, and Word would **not** 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 to click "Enable" will make ALL the difference in thwarting spoofing attacks.



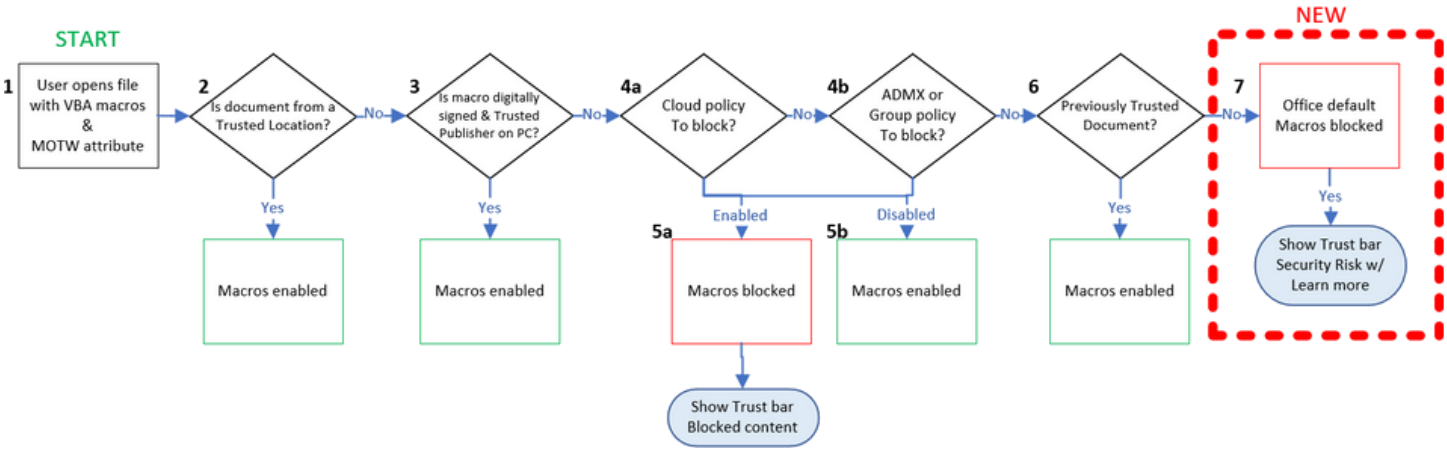
Microsoft said that at a future date to be determined, they also plan to make this change to Office LTSC, Office 2021, Office 2019, Office 2016, and Office 2013.

So, whereas before the bar said "Security Warning" with a yellow exclamation point, the new bar says "Security Risk" with a red "X" and the further explanation: "Microsoft has blocked macros from running because the source of this file is untrusted."

This will, without question, put a serious kink in the capabilities of malware gangs who have 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 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, malicious payloads are delivered, and the impact can be severe including malware, compromised identity, data loss, and remote access."*

You're just figuring this out now, Microsoft? But I'm not looking a gift horse in the mouth. Better late than never. The logic tree gauntlet that macros will need to run finally gives them the respect their power should have always commanded.



"MOTW" shown in the initial box stands for "mark of the web" which is a flag Microsoft automatically tags files with which have come from the Internet. I'm sure that our listeners will have seen those pop-up warnings when Windows is aware that a file came from the Internet. That's the Mark of the Web.

Now, it also turns that most of us have been victims of "The Tyranny of the Default." since, believe it or not, there's always been a group policy 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. 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.

Never11?

For many months I've been ignoring a continual stream of questions asking whether, and hopefully when, I would be offering a "Never11" utility, similar to the "Never10" utility that has become a bit famous. At the moment, none of my own systems qualify to move to Windows 11. But we know that's 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: "AllowUpgradesWithUnsupportedTPMOrCPU." So, they'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.

And as for Windows 11, 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 to until they put back a bunch of the features they've taken away from Windows 10. 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, 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 not sound 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. Period. It is, of course, not enabled by default. But it is 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 that she assumed not, but that she would place a formal request with Microsoft to ask. I worked on it Friday and by the end of the evening I posted a test release of Never11 for GRC's newsgroup gang to play with. 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 is what I want to be working on. 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 soon... and I should have something new, fun and useful to announce next week.

The Inept Panda

So I think that the best way for me to begin will be to first 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.

Citizen Lab is an interdisciplinary laboratory based at the Munk School of Global Affairs & Public Policy at the University of Toronto, 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 of popular applications, and examining transparency and accountability mechanisms relevant to the relationship between corporations and state agencies regarding personal data and other surveillance activities.

In other words, these guys are perfectly positioned to be interested in understanding the detailed operation of the smartphone app that all 2022 Olympic athletes and attendees are being required to install and use.

They begin their report with a bit of background to set the stage, writing:

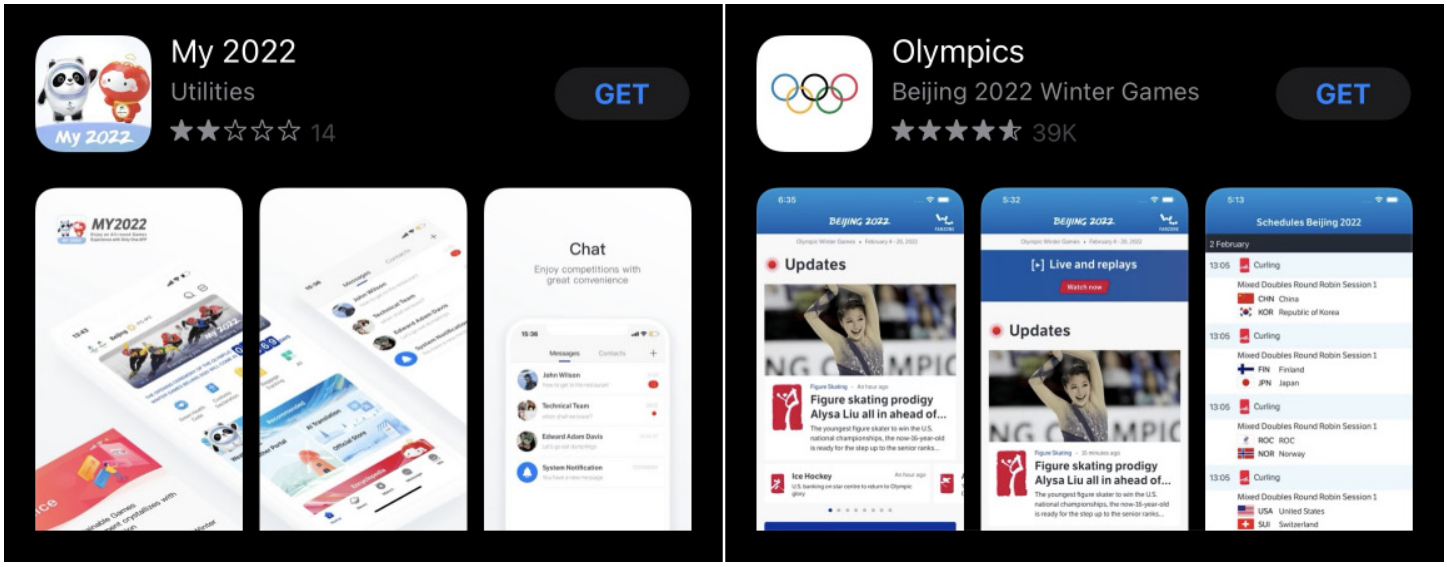
> *The 2022 Winter Olympic Games in Beijing have generated significant controversy. As early as February 2021, over 180 human rights groups had called for governments to boycott the Olympics, arguing that holding the Games in Beijing will legitimize a regime currently engaging in genocide against Uyghur 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 their own human rights commitments by cooperating with the Chinese government. Following a professional tennis player's 2021 sexual assault accusation against a Chinese Communist Party leader and her subsequent disappearance, Human Rights Watch stated that "the IOC has vaulted itself from silence about Beijing's abysmal human rights record to active collaboration with Chinese authorities in undermining freedom of speech and disregarding alleged sexual assault." According to IOC documents, the accused Chinese Communist Party leader headed the steering committee charged with securing and organizing the 2022 Games.*

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 blocklist of 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.

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. 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.

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 that app and another generically titled "Olympics":



The official Chinese "MY2022" had 14 reviews averaging 2 stars whereas the generic "Olympics" app had 39 thousand reviews averaging just shy of a full 5 stars. Looks like about 4.75 out of 5.

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:

- MY2022, an app mandated for use by all attendees of the 2022 Olympic Games in Beijing, has a simple but 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.

- 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 is unclear with whom or which organization(s) it shares this information.

- 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 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.

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 personal identifiable information including users' demographic information and passport information (i.e., issue and expiration dates) as well as the organization to which the individual belongs.

The official Olympics 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, COVID-19 vaccination status, and COVID-19 lab test results for such purposes. While the official Olympics 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," MY2022's privacy policy itself did not specify with whom or with which organization(s) it would share users' medical and health-related information.

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, 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.

No 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 connected 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 meaningless. Verifying certificate authenticity is so crucial that both the iOS and Android platforms 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 whose certificates are not being checked seem quite important:

- my2022.beijing2022.cn
- tmail.beijing2022.cn
- dongaoserver.beijing2022.cn
- app.bcia.com.cn
- health.customsapp.com

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 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 authentication has been all over the news since Citizen Labs' report went public exactly three weeks ago. 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 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.

Being responsible, and just hoping to be able to get these important oversights fixed before the Games began, Citizen Lab privately disclosed their findings to the Beijing Organizing Committee for the 2022 Olympic and Paralympic Winter Games back at the beginning of December, on the 3rd, of 2021. 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.

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, one 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 Labs' 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 chit chat, Citizen Lab promptly examined the update application.

Not only had none of the known 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, while 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.

The Censorship side...

According to MY2022's description in Apple's App Store, the app implements a wide range of communication functionalities including real-time chat, news feeds, 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 these keywords were 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.) 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 be called. So, who knows? We'll never know.

A spokesman for the International Olympic Committee justified the app's security issues — apparently without understanding them at all — by saying that due to the COVID-19 pandemic, "special measures" needed to be put in place. What?!?! This individual also defended the app by saying it received approval from the Google Play store and the App Store. Oh well, 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… The 'My2022' app supports the function for health monitoring. It is designed to keep Games-related personnel safe within the closed-loop environment."

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