Security Now! #803 - 01-26-21

Comparative Smartphone Security

This week on Security Now!

This week we look at the updates in release 88 of both Chrome and Edge with their evolving password manager features. We also look at two recent head-shaking consequences of the hard end-of-life for Adobe's Flash. Ransomware gangs have added another new incentive for payment, and additional details continue emerging about last year's SolarWinds attacks. We have newly disclosed discoveries from a Google Project Zero researcher and I spend a bit of time wondering out loud how we're ever going to change the low priority that's currently being given to serious security problems that don't directly inconvenience end users. And we finish by examining a very useful analysis of the comparative security of iOS and Android recently published by Johns Hopkins' Matthew Green and team.

The Unfortunate Reality of Perverse Incentives

As we'll see, this ties in a bit with the worry over how we're ever going to improve the security of our systems and devices when the economic incentives to actually do so do not exist.

Browser News

Browser Password Managers

During this past week both Chrome and Edge have beefed-up their browser's built-in password managers.

Chrome 88 was released a week ago on the 19th. And despite using Chrome daily, it wasn't until I went to "About Chrome" that it was induced to move from 87 to 88 and then restart. One of the nice things about Chrome's password system is its quick easy of access. You can use the menu system to search for passwords. But passwords occurs inn multiple places. So that's less than ideal. Or, you can use the clear and clean URL: "chrome://settings/passwords". But the niftiest and easiest-to-remember way to get there is to click on your circular logged-in picture or avatar in the upper right hand corner and you'll find a little skeleton key as the leftmost icon directly beneath your name and eMail address. Clicking it takes you directly to Chrome's password management page — the same place that //settings/passwords takes you.

Google's Security Blog page is titled: "New Year, new password protections in Chrome"
https://security.googleblog.com/2021/01/new-year-new-password-protections-in.html

Under "Easily fix weak passwords":
In Chrome 88, you can now complete a simple check to identify any weak passwords and take action easily.

Under "Edit your passwords in one place":
Chrome can already prompt you to update your saved passwords when you log in to websites. However, you may want to update multiple usernames and passwords easily, in one convenient place. Starting in Chrome 88, you can manage all of your passwords faster and easier in Chrome Settings on desktop and iOS (Chrome's Android app will be getting this feature soon, too).

They finished by saying:
"The new features with Chrome 88 will be rolled out over the coming weeks, so take advantage of the new updates to keep your passwords secure. Stay tuned for more great password features throughout 2021."

It's unclear what isn't already working today since everything appears to be present and working.

Meanwhile, Microsoft is also announcing new password features in Edge 88.
https://docs.microsoft.com/en-us/deployedge/microsoft-edge-relnote-stable-channel

In their "Release notes for Microsoft Edge Stable Channel" announcement for 88.0.705.50, under the topic of passwords, Microsoft said:

Password Generator. Microsoft Edge offers a built-in strong password generator that you can use when signing up for a new account or when changing an existing password. Just look for the browser-suggested password drop down in the password field and when selected, it will automatically save to the browser and sync across devices for easy future use.

Password Monitor. When any of your passwords saved to the browser matches with those seen in the list of leaked credentials, Microsoft Edge will notify you and prompt you to update your password. Password Monitor scans for matches on your behalf and is on by default.

Edit Password. You can now edit your saved passwords directly in Microsoft Edge Settings. Any time a password has been updated outside of Microsoft Edge, it's easy to replace the saved older password with the new one by editing the saved entry in Settings.

So, what we're seeing with all of the major browsers is the inevitable addition of native password management. Given how long we've been happily using LastPass, the only question I have is what took them so long. And, of course, the trouble with using any one make of browser's built-in password manager, is the loss of cross-browser usage. I'm certain that this inherent "lock-in" is seen as a big advantage by the various browser publishers, which is exactly why I will be continuing to rely upon a 3rd-party browser extension.

Other browser updates:
Also in both of the 88's — Chrome and Edge, FTP client support has finally been shutdown, and all support for Adobe's Flash player and format have been removed.

For those who live in the dark, Chrome has also improved their support for Dark Mode. Before 88, the support had been incomplete with some controls not having dark mode treatments. But with 88, Chrome becomes a fully resurfaced Dark Mode client.

That "Incremental Search Within Tabs" feature that we talked about before is here, now in Chrome, even though it will probably be disabled at first. It can be enabled at: chrome://flags/#enable-tab-search.

There are also a bunch of developer-side enhancements and additions, and unspecified security updates.

And, finally, this release 88 of Edge is supposed to be the one to, at long last, give us a built-in option for vertical tabs. But I went looking for the option to enable it on the Appearance page where there's supposed to be an option to display the Vertical Tabs switching icon to the left of the tab row. But so far it's not giving me the option.

Flash

The Trains Run On Time — but not in China.
Speaking of the end of Adobe Flash, a wonderful news report's headline reads: *"Adobe Flash Shutdown Halts Chinese Railroad for Over 16 Hours Before Pirated Copy Restores Ops"* with the subhead: "This is what happens when you RUN A RAILROAD NETWORK ON FLASH."

https://www.thedrive.com/news/38897/adobe-flash-shutdown-halts-chinese-railroad-for-over-16-hours-before-pirated-copy-restores-ops

The story was published on “The Drive” website which describes itself as “the one-stop shop for all things automotive” and was written by James Gilboy (@_JamesGilboy) who clearly knows a bit of the tech and security history of Flash:

> Supportive of everything from browser games to live streaming, Adobe Flash was the internet's favorite multimedia platform with reason. Even in its heyday, though, Flash wasn't universally loved; it had security holes, could be tough to optimize, and wouldn't play ball with all browsers, especially those on mobile devices. When HTML5 hit the scene, Flash began to fall out of favor, and in July 2017, Adobe announced it would cease support at the end of 2020, giving users three and half years to switch to new software. This message, however, didn't reach all corners of the IT globe, and when Flash's "time bomb" code went off on January 12, it did more than just make nostalgic browser games harder to revisit: It brought an entire Chinese railroad to a standstill.
>
> According to a report by Apple Daily, the problem reared its head for China Railway Shenyang in Dalian, Liaoning just after 8:00 a.m. on Tuesday, Jan. 12. Per an event timeline outlined by Github, the head of a switching station reported being unable to access the railroad's timetables, which they normally did through a browser-based Flash interface. Over the next half hour, reports of similar failures poured in from across the network, with as many as 30 stations implicated according to a CR Shenyang statement reported by a Chinese blog.
>
> Only after technicians went online to research bug fixes did officials learn of the global Flash shutdown, news of which seemingly didn't penetrate the insular Chinese internet. A translation of the Github timeline suggests restoring software backups temporarily restored service around noon, though outages returned again at around 2:00 p.m., and later on in the evening. CR Shenyang's response team then reportedly began exploring a reversion to older software systems, its options apparently consisting of an unspecified Microsoft-based setup, or an archived, pirated version of Flash without the "time bomb" code. Technicians settled on the latter, and around 1:00 a.m. on the 13th, CR Shenyang successfully brought one of its stations fully online. By 2:30 a.m., all but one route was back in service and the railroad's Y2K21 nightmare behind it.

The SARS Browser (South African Revenue Service)

Google’s well-known software engineer, Ryan Sleevi, tweeted:

> Ryan Sleevi
> @sleevi_
> South African Revenue Service decides to develop and distribute their own browser, specifically to re-enable Adobe Flash. 😳

https://tools.sars.gov.za/webtools/sarsbrowser/browserdownload.aspx

https://www.sars.gov.za/AllDocs/OpsDocs/Guides/GEN-ELEC-21-G01%20-%20How%20to%20download%20the%20new%20SARS%20eFiling%20Browser%20-%20External%20Guide.pdf

Dear Taxpayers and Traders,

SARS apologises for the inconvenience and service disruption caused by the discontinuance of the Adobe Flash player.

We are pleased to inform you that an alternate SARS Browser solution has been implemented which affords you the ability to complete and submit the Flash-based forms not migrated to HTML5, in the interim, while we complete the migration.

The SARS Browser enables access to ALL eFiling forms, including those that require Adobe Flash, thus maintaining compliance with your filing obligations.

Please note that existing browsers such as Chrome & Edge will continue to work for all forms already migrated with the major and high volume ones being Income Tax (PIT, Provisional Tax, CIT & Trusts), Value Added Tax, Pay as you Earn and Excise.

You are please requested to use the SARS browser should access to the forms not yet migrated be required, which include:

- RAV01 Registration, Amendments and Verification Form
- TDC01 Transfer Duty
- IT3-01 Financial Certificate Information
- IT3-02 Financial Declaration
- TCR01 Tax compliance Status Request
- DTR01 Dividends Tax Transactions Information
- WTI Withholding Tax on Interest

Please note that the SARS Browser will require software to be installed on your PC and is currently compatible with Windows devices only.

This SARS browser deploys as a separate application and can only be used to access the SARS eFiling website and SARS Corporate website. It cannot be used as a browser for general internet browsing.

Setting aside the unfortunate abbreviation of the South African Revenue Service (SARS), what this means is that lord only knows how many years ago, South Africa's taxing authority implemented their Internet browser-based tax eFiling system in Flash. It's not clear why, even then, that was a sane choice. But I suppose that not even JavaScript was ready for prime time back when Flash was offering a powerful client-side scripting environment. So if one wanted to be performing on-the-fly form content validation before submission, offering drop-down lists, checkboxes and such, with some local intelligence, Flash may have been the best solution, then.

So this is another perfect example of how very difficult it is to change things that are working. Everyone is busy. And despite the fact that Abode announced the end-of-life for Flash back in 2017, the SARS IT people were nevertheless caught off guard two weeks ago with the discovery that Adobe had built-in a time bomb that would cause Flash to refuse to run after January 1th. The last time-bomb free version of Flash was 32.0.0.363. Versions 32.0.0.371 and on all contain the time-bomb. There is work on Github to remove the time-bomb, but browsers are also now refusing to host Flash, so it really is time to move on.

Ransomware News

Post Ransomware DDoS

Now that the threat of an expensive ransomware attack is quite real, investments have been made in the ability to restore from backups. So today, more than just a few years ago, when a company suffers a ransomware attack, many more are able to restore from backups and thus avoid the need to contact the attackers.

The first response to the lack of need to pay the ransom was to exfiltrate files, release a few as proof of possession, and threaten to release many more if a ransom was not paid.

But if the offer to decrypt encrypted files and an agreement to never leak and to destroy sensitive and perhaps personal information are insufficient to bring a company and its C-suite executives to heel, how about subjecting the organization to a prolonged DDoS attack?

That's the latest attack strategy being added to post-ransomware attacks: Blast'em off the Internet until they agree to cooperate and pay. This new trend was first seen last October, 2020. It was being employed by the SunCrypt and RagnarLocker gangs after their attacks. But now a third player, the Avaddon ransomware gang, has begun using DDoS attacks to take down a victim's site and network until the victim contacts them and begins negotiating.

Brett Callow, the threat analyst for Emsisoft, said: “It’s not at all surprising to see threat actors combining ransomware and DDoS attacks: DDoS is cheap, easy and in some cases may help convince some companies that speedy payment is the least painful option. The more pressure the criminals can put companies under, the better their chances of extracting payment.”

Security News

SolarWinds attack details continue to emerge.

As we know, digital attack forensics takes time. And most of it requires reverse engineering code that has been carefully designed to thwart exactly that sort of analysis. So we would expect to be learning more about this largest known attack in history over time. And, indeed, last Wednesday the 20th, in a joint posting by the Microsoft 365 Defender Research Team, the Microsoft Threat Intelligence Center (MSTIC), and the Microsoft Cyber Defense Operations Center (CDOC), we learned a great deal more. And for what it's worth, it is only more worrisome.

https://www.microsoft.com/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop/

Microsoft's joint disclosure was titled: “Deep dive into the Solorigate second-stage activation: From SUNBURST to TEARDROP and Raindrop”

Microsoft begins their quite lengthy disclosure with a brief summary of what everyone knows, but quickly begins to add new detail. They wrote:

More than a month into the discovery of Solorigate, investigations continue to unearth new details that prove it is one of the most sophisticated and protracted intrusion attacks of the decade. Our continued analysis of threat data shows that the attackers behind Solorigate are skilled campaign operators who carefully planned and executed the attack, remaining elusive while maintaining persistence. These attackers appear to be knowledgeable about operations security and performing malicious activity with minimal footprint. In this blog, we'll share new information to help better understand how the attack transpired. Our goal is to continue empowering the defender community by helping to increase their ability to hunt for the earliest artifacts of compromise and protect their networks from this threat.

We have published our in-depth analysis of the Solorigate backdoor malware (also referred to as SUNBURST by FireEye), the compromised DLL that was deployed on networks as part of SolarWinds products, that allowed attackers to gain backdoor access to affected devices. We have also detailed the hands-on-keyboard techniques that attackers employed on compromised endpoints using a powerful second-stage payload, one of several custom Cobalt Strike loaders, including the loader dubbed TEARDROP by FireEye and a variant named RAINDROP by Symantec.

One missing link in the complex Solorigate attack chain is the handover from the Solorigate DLL backdoor to the Cobalt Strike loader. Our investigations show that the attackers went out of their way to ensure that these two components are separated as much as possible to evade detection. This blog provides details about this handover based on a limited number of cases where this process occurred. To uncover these cases, we used the powerful, cross-domain optics of Microsoft 365 Defender to gain visibility across the entire attack chain in one complete and consolidated view.

We'll also share our deep dive into additional hands-on-keyboard techniques that the attackers used during initial reconnaissance, data collection, and exfiltration, which complement the broader TTPs from similar investigative blogs, such as those from FireEye and Volexity.

Then, a bit later, they explain:

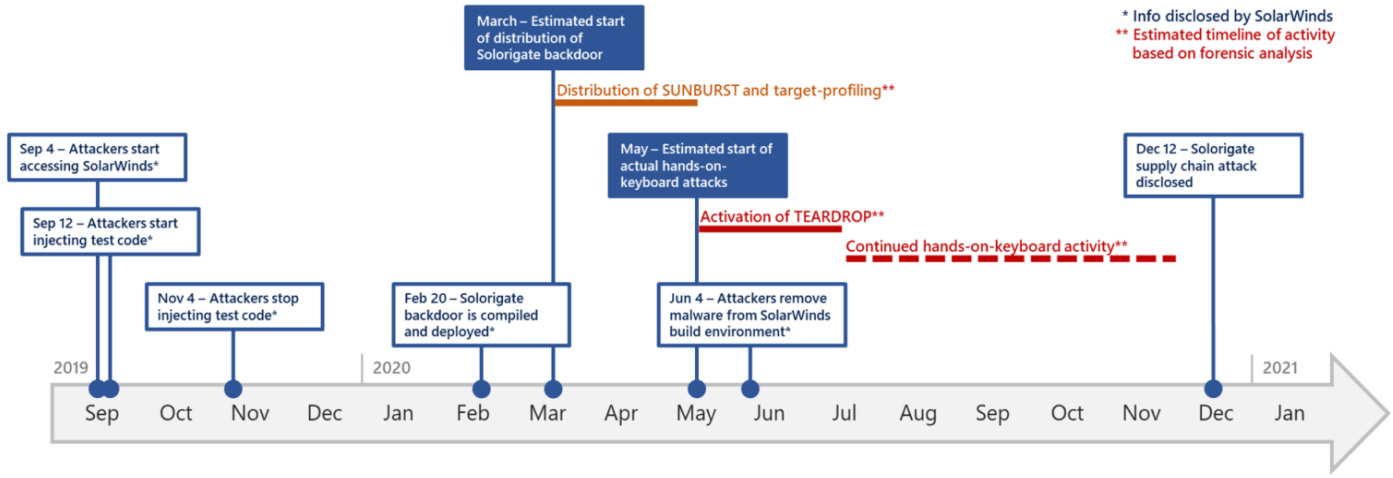
An attack timeline that SolarWinds disclosed in a recent blog showed that a fully functional Solorigate DLL backdoor was compiled at the end of February 2020 and distributed to systems sometime in late March. The same blog also said that the attackers removed the Solorigate backdoor code from SolarWinds' build environment in June 2020.

Considering this timeline and the fact that the Solorigate backdoor was designed to stay dormant for at least two weeks, we approximate that the attackers spent a month or so in selecting victims and preparing unique Cobalt Strike implants as well as command-and-control (C2) infrastructure. This approximation means that real hands-on-keyboard activity most likely started as early as May.

The removal of the backdoor-generation function and the compromised code from SolarWinds binaries in June could indicate that, by this time, the attackers had reached a sufficient number of interesting targets, and their objective shifted from deployment and activation of

the backdoor (Stage 1) to being operational on selected victim networks, continuing the attack with hands-on-keyboard activity using the Cobalt Strike implants.

That was news to me. I always assumed that the SolarWinds build and update delivery system had remained infected. But not so. As Microsoft observed, it didn't need to keep offering infected DLL's once all of the major targets had updated and received the infection.



One of the coolest things Microsoft found was the way the original SolarWinds infection created an "arm's length" execution path in such a way that the original infection stood a maximum chance of remaining undetected even if its consequences were. Remember that the moment it was discovered that a signed SolarWinds DLL was the root source of the infection, that would bring down the entire operation. As we know, that **is** what eventually happened at FireEye, but the obfuscation was successful for a **very** long time. Here's now Microsoft explains what they found:

We spent countless hours investigating Microsoft Defender telemetry and other signals from potential patient-zero machines running the backdoored version of SolarWinds DLL. Most of these machines communicated with the initial randomly generated DNS domain .avsvmcloud.com but without significant activity. However, we saw limited cases in May and June where the initial DNS network communication was closely followed by network activity on port 443 (HTTPS) to other legit-looking domains. On these handful of machines, we performed deep inspection of telemetry.

We know that the Solorigate backdoor only activates for certain victim profiles, and when this happens, the executing process (usually SolarWinds.BusinessLayerHost.exe) creates two files on disk:

- A VBScript, typically named after existing services or folders to blend into legitimate activities on the machine

- A second-stage DLL implant, a custom Cobalt Strike loader, typically compiled uniquely per machine and written into a legitimate-looking subfolder in %WinDir% (e.g., C:\Windows)

At this point the attackers are ready to activate the Cobalt Strike implant. However, the attackers apparently deem the powerful SolarWinds backdoor too valuable to lose in case of discovery, so they tried to separate the Cobalt Strike loader's execution from the SolarWinds process as much as possible. Their hope is that, even if they lose the Cobalt Strike implant due to detection, the compromised SolarWinds binary and the supply chain attack that preceded it are not exposed.

The attackers achieved this by having the SolarWinds process create an Image File Execution Options (IFEO) Debugger registry value for the process dllhost.exe. [This is a known and official way of attaching a Windows debugger to a process at startup. So it's a very clever repurposing of a Windows feature by the malware.] Once the registry value is created, the attackers simply wait for the occasional execution of dllhost.exe, which will typically happen naturally on a system. This execution triggers a process launch of wscript.exe configured to run the VBScript file dropped earlier and has been waiting.

The VBScript in turn runs rundll32.exe, activating the Cobalt Strike DLL using a clean parent/child process tree completely disconnected from the SolarWinds process. Finally, the VBScript removes the previously created IFEO value in the registry to clean up any traces of execution.

Microsoft's full posting is super long and wonderfully detailed for anyone wanting to really get down into the details. But in the section "Additional attacker tactics, anti-forensic behavior, and operational security" Microsoft nicely summarizes some more detail to give us a more complete sense for the frightening skill and tradecraft that the designers of this attack employed:

As mentioned, the attackers behind Solorigate are skillful and methodic operators who follow operations security (OpSec) best practices to minimize traces, stay under the radar, and avoid detection.

During our in-depth analysis of the attacker's tactics, techniques, and procedures (TTPs) seen through the lens of Microsoft 365 Defender's rich telemetry, we observed a few techniques that are worth disclosing to help other defenders better respond to this incident and use hunting tools like Microsoft 365 Defender advanced hunting or Azure Sentinel queries to search for potential traces of past activity.

Some examples of why these attackers stand out for their professional OpSec methodology and anti-forensic behavior are listed below:

- Methodic avoidance of shared indicators for each compromised host. As discussed in the previous section, each Cobalt Strike DLL implant was prepared to be unique per machine and avoided at any cost overlap and reuse of folder name, file name, export function names, C2 domain/IP, HTTP requests, timestamp, file metadata, config, and child process launched. This extreme level of variance was also applied to non-executable entities, such as WMI persistence filter name, WMI filter query, passwords used for 7-zip archives, and names of output log files. Applying this level of permutations for each individual compromised machine is an incredible effort normally not seen with other adversaries and done to prevent full identification of all compromised assets inside a network or effective sharing of threat intel between victims.

- Camouflage and blending into the environment. Tools and binaries used by the attackers (e.g., ADFIND legit tool) were always renamed and placed in folders that mimicked existing programs and files already present on a machine. This blending was not just used for files, but for other elements. For example, WMI persistence filters were created with names and queries matching other scripts present in affected organizations.

- Before running intensive and continued hands-on keyboard activity, the attackers took care of disabling event logging using AUDITPOL and re-enabling it afterward.

- In a similar way, before running noisy network enumeration activities (such as repeated NSLOOKUP or LDAP queries), the attackers carefully prepared special firewall rules to minimize outgoing packets for certain protocols. The firewall rules were also methodically removed after the network reconnaissance was completed.

- Lateral movement activities were never executed without preparation. To increase the likelihood that their activities remain undetected, the attackers first enumerated remote processes and services running on the target host and decided to move laterally only after disabling certain security services.

- We believe that the attackers used timestomping to change timestamps of artifacts and also leveraged professional wiping procedures and tools to complicate finding and recovering of DLL implants from affected environments.

Stepping back to take stock in all that we have learned, any sane InfoSec technologist would be right to be seriously worried. My feeling is that as damaging as these attacks were individually on their own, it's almost a worse outcome for the attackers for us to have obtained this much greater appreciation for their skill and dedication to detail.

And none of us should forget for a moment that were it not for the fact that they targeted FireEye, and that their presence eventually tripped some monitoring alarms that the attackers were unaware of... this would all still be ongoing right now.

Malwarebytes was also attacked

Last Tuesday Malwarebytes posted a notice with the headline: "Malwarebytes targeted by Nation State Actor implicated in SolarWinds breach. Evidence suggests abuse of privileged access to Microsoft Office 365 and Azure environments"

https://blog.malwarebytes.com/malwarebytes-news/2021/01/malwarebytes-targeted-by-nation-state-actor-implicated-in-solarwinds-breach-evidence-suggests-abuse-of-privileged-access-to-microsoft-office-365-and-azure-environments/

I want to share their short summary posting because it's impressive, it contains additional important details, and also helps to highlight the nature and need for a true security community:

A nation state attack leveraging software from SolarWinds has caused a ripple effect

throughout the security industry, impacting multiple organizations. We first reported on the event in our December 14 blog and notified our business customers using SolarWinds asking them to take precautionary measures.

While Malwarebytes does not use SolarWinds, we, like many other companies were recently targeted by the same threat actor. We can confirm the existence of another intrusion vector that works by abusing applications with privileged access to Microsoft Office 365 and Azure environments. After an extensive investigation, we determined the attacker only gained access to a limited subset of internal company emails. We found no evidence of unauthorized access or compromise in any of our internal on-premises and production environments.

How did this impact Malwarebytes?

We received information from the Microsoft Security Response Center on December 15 about suspicious activity from a third-party application in our Microsoft Office 365 tenant consistent with the tactics, techniques and procedures (TTPs) of the same advanced threat actor involved in the SolarWinds attacks.

We immediately activated our incident response group and engaged Microsoft's Detection and Response Team (DART). Together, we performed an extensive investigation of both our cloud and on-premises environments for any activity related to the API calls that triggered the initial alert. The investigation indicates the attackers leveraged a dormant email protection product within our Office 365 tenant that allowed access to a limited subset of internal company emails. We do not use Azure cloud services in our production environments.

Considering the supply chain nature of the SolarWinds attack, and in an abundance of caution, we immediately performed a thorough investigation of all Malwarebytes source code, build and delivery processes, including reverse engineering our own software. Our internal systems showed no evidence of unauthorized access or compromise in any on-premises and production environments. Our software remains safe to use.

What we know: SolarWinds Attackers Also Target Administrative and Service Credentials

As the US Cybersecurity and Infrastructure Security Agency (CISA) stated, the adversary did not only rely on the SolarWinds supply-chain attack but indeed used additional means to compromise high-value targets by exploiting administrative or service credentials.

In 2019, a security researcher exposed a flaw with Azure Active Directory where one could escalate privileges by assigning credentials to applications. In September 2019, he found that the vulnerability still existed and essentially lead to backdoor access to principals' credentials into Microsoft Graph and Azure AD Graph.

Third-party applications can be abused if an attacker with sufficient administrative privilege gains access to a tenant. A newly released CISA report reveals how threat actors may have obtained initial access by password guessing or password spraying in addition to exploiting administrative or service credentials. In our particular instance, the threat actor added a self-signed certificate with credentials to the service principal account. From there, they can

authenticate using the key and make API calls to request emails via MSGraph.

For many organizations, securing Azure tenants may be a challenging task, especially when dealing with third-party applications or resellers. CrowdStrike has released a tool to help companies identify and mitigate risks in Azure Active Directory.

Coming together as an industry

While we have learned a lot of information in a relatively short period of time, there is much more yet to be discovered about this long and active campaign that has impacted so many high-profile targets. It is imperative that security companies continue to share information that can help the greater industry in times like these, particularly with such new and complex attacks often associated with nation state actors.

We would like to thank the security community, particularly FireEye, CrowdStrike, and Microsoft for sharing so many details regarding this attack. In an already difficult year, security practitioners and incident responders responded to the call of duty and worked throughout the holiday season, including our own dedicated employees. The security industry is full of exceptional people who are tirelessly defending others, and today it is strikingly evident just how essential our work is moving forward.

As if we didn't already have sufficient cause to be worried about the SolarWinds threat actor, now we learn that this incredibly potent adversary is not a one trick pony. Not only were they also mounting this other entirely different attack, they were doing so at the same time as they were carefully and delicately rooting around within the world's highest-end corporate and government networks.

It seems that wherever we look we find problems

We'll all recall the critical flaw that was found in Apple's implementation of FaceTime early in 2019. The flaw made it possible for users to initiate a FaceTime video call and eavesdrop on their callees by adding their own number as a third person in a group chat even before the person on the other end accepted the incoming call. Apple immediately took FaceTime's group chat feature offline until the oversight was fixed, as it was in a subsequent update.

The interesting nature of this flaw captured the attention of Google Project Zero researcher Natalie Silvanovich. She describes what made this flaw so different and interesting, as follows:

On January 29, 2019, a serious vulnerability was discovered in Group FaceTime which allowed an attacker to call a target and force the call to connect without user interaction from the target, allowing the attacker to listen to the target's surroundings without their knowledge or consent. The bug was remarkable in both its impact and mechanism. The ability to force a target device to transmit audio to an attacker device without gaining code execution was an unusual and possibly unprecedented impact of a vulnerability. Moreover, the vulnerability was a logic bug in the FaceTime calling state machine that could be exercised using only the user interface of the device. While this bug was soon fixed, the fact that such a serious and easy to

reach vulnerability had occurred due to a logic bug in a calling state machine -- an attack scenario I had never seen considered on any platform -- made me wonder whether other state machines had similar vulnerabilities as well. This post describes my investigation into calling state machines of a number of messaging platforms, including Signal, JioChat, Mocha, Google Duo, and Facebook Messenger.

https://googleprojectzero.blogspot.com/2021/01/the-state-of-state-machines.html

In her very detailed posting, Natalie proceeds to explain how WebRTC sessions are established between caller and callee, and how, unless particular care is taken in the interactive handshaking between the endpoints, a number of exploits are possible — arising from subtle mistakes made in the implementation of the governing protocol's state machines. For anyone who's interested in the details, the link to Natalie's posting is in the show notes. So I'll only share what Natalie's subsequent research uncovered:

As with Apple's pre-patched FaceTime, many other apps allowed calls to be connected without interaction from the callee, while also potentially permitting the caller to force a callee device to transmit audio or video data. As noted above, the common root cause were logic bugs within the signaling state machines, which Natalie described as a "a concerning and under-investigated attack surface of video conferencing applications."

So, what did she find?

- Signal (fixed in September 2019) - A audio call flaw in Signal's Android app made it possible for the caller to hear the callee's surroundings due to the fact that the app didn't check if the device receiving the connect message from the callee was the caller device.

- Both JioChat (fixed in July 2020) and Mocha (fixed in August 2020) - Adding candidates to the offers created by Reliance JioChat and Viettel's Mocha Android apps that allowed a caller to force the target device to send audio (and video) without a user's consent. The flaws stemmed from the fact that the peer-to-peer connection had been set up even before the callee answered the call, thus increasing the "remote attack surface of WebRTC."

- Facebook Messenger (fixed in November 2020) - A vulnerability that could have granted an attacker who is logged into the app to simultaneously initiate a call and send a specially crafted message to a target who is signed in to both the app as well as another Messenger client such as the web browser, and begin receiving audio from the callee device.

- Google Duo (fixed in December 2020) - A race condition between disabling the video and setting up the connection that, in some situations, could cause the callee to leak video packets from unanswered calls.

I titled this piece "It seems that wherever we look we find problems" because that seems to be today's reality. Natalie wasn't driven to examine these communication applications because of any known flaws. She was just wondering whether anything **might** be wrong. And she discovered significant logical flaws enabling their use for surreptitious surveillance.
The complexity of today's apps is exploding. The bar for what an app must do to competitively

succeed keeps being raised. Those raising it are throwing together huge functional blocks of code without the deep understanding required to consider — or even be aware of — the security implications created by these massive conglomerations.

Need real time audio and video chat? No problem, just drop in and hook up a WebRTC library. Wave at the camera. Can you see yourself at the other end? Yes? Great! We're done! Ship it.

And when one app does this, those competing with that app are forced to follow suit, because reviewers never see the logic flaws that interested Natalie. Reviews create app feature grids filled with red and green squares, showing which apps offer which features. So anyone whose app has a glaring red box highlighting a missing feature is in a big hurry to go grab some barely understood 3rd-party library, drop it in, hook it up, wave at the camera, and push out a new feature release.

And I'm not sure how or why this is ever going to change. Some things do change. I still have the "Ctrl-S" habit, developed from the need to constantly save my work, because back at the dawn of all this, you never knew when the app or operating system you were using was going to freeze up, which was something that generally happened multiple times per day without warning. That doesn't generally happen anymore. Not like it once did. But the problem is, everything is driven by economics. When apps and OSes crashed, and work was lost, users were unhappy. This created an economic cost because there was an incentive to seriously consider trying a different app or OS. Consequently, apps and OSes no longer crash like they did.

But not one of the users of these audio and video conferencing apps were inconvenienced by the logical flaws in their internal state machines which enabled eavesdropping. They might be annoyed if the app didn't offer the "turn my face into an elf" option... so, of course, every app needs to offer that, or suffer the red grid cell.

So I think we're in trouble because users and reviewers do not and cannot perceive security flaws. They only see features. IoT devices pay no economic toll for using a TCP/IP stack that's riddled with critical flaws, nor for not providing any aftermarket means for updating them. Doing so would be expensive. But no rational product designer in China is going to invest in anything that doesn't provide an economic advantage.

I see three possibilities.

Back in the early days of electrical household appliances which were playing with dangerous voltages and currents, someone needed to establish a standard for things like the thickness of insulation and the presence of a grounded plug to minimize shock hazard in the event of an internal failure. As with today's computer security, consumers were unable to evaluate the safety — or lack thereof — that had been designed into their toasters and vacuum cleaners. Consequently, there was no economic benefit associated with adding manufacturing cost to make appliances more safe. Insulation costs money. A grounded three-wire cord is more expensive than one with just two wires. Things were a mess until Underwriters Laboratories with their "UL Seal of Approval" became a household term. Now consumers could look for a **UL** seal to assure them that some white lab-coated scientist had dropped the toaster they were considering purchasing into a tub of water to see what it would happen. And if that toaster did the right thing, it would get the seal... and the similar-looking but less expensive toasters next

to it on the same shelf — but without the UL seal — would go unsold. Since there was no way Mom was going to expose little Johnny and Suzie to any dangerous appliances, the **UL** seal was soon effectively required, and all toasters had to invest in the additional manufacturing cost to earn the seal.

Later, the famous "Consumer Reports" served a similar role. Consumers who took one look under the hood of their car quickly realized that they were unable to evaluate what they saw. Nor were they able to perform their own crash testing. So they needed to rely upon a neutral highly reputable independent testing organization to help them choose.

So, perhaps something like Underwriters Laboratories or Consumer Reports will emerge. But the other inevitability, I think, is the heavy hand of government regulation. Seatbelts do save lives. Many people object to being told to buckle up. And no automaker wanted to put them in. There was no obvious economic benefit for doing so, and even a liability if the belt appeared to malfunction when it was needed. And automotive pollution standards fall under the same category. They are good for society collectively, though expensive for automakers and consumers individually. If given a choice, pollution standards would never have occurred voluntarily. The government needed to force it. Anyone who remembers what the air in Los Angeles once looked like — or what Bejing looked like more recently — can appreciate that sometimes government regulation against short-term economic interest is required.

SolarWinds has been a massive wake-up call to many of the world's government legislators. But it inherently targeted and affected large public and private organizations. More than anything, it demonstrates the level of malign intent and energy that is focused against the US and other international organizations. I would not be surprised if we eventually discover that something similar has been going on with some subset of the hundreds of millions of US consumer networks that are increasingly hosting IoT devices with unknown security flaws. That may be the wake up call we need to get serious about the security of our devices.

Sci-Fi

"The Expanse"

Comparative Smartphone Security

A team of security researchers at Johns Hopkins, led by cryptographer, security technologist and associate professor of computer science, our friend Matthew Green, decided to take a serious look at the comparative security offered by the Apple iOS vs Google Android smartphone platforms.

To host the results of their analysis they grabbed the domain securephones.io.

https://securephones.io/ https://securephones.io/main.html https://securephones.io/main.pdf

Here's how they framed the intent, goals and very brief summary of their research:

> In this work we present definitive evidence, analysis, and (where needed) speculation to answer the questions:
>
> 1. "Which concrete security measures in mobile devices meaningfully prevent unauthorized access to user data?"
> 2. "In what ways are modern mobile devices accessed by unauthorized parties?" and finally,
> 3. "How can we improve modern mobile devices to prevent unauthorized access?"
>
> We divide our attention between two major platforms in the mobile space, iOS and Android, and for each we provide a thorough investigation of existing and historical security features, evidence-based discussion of known security bypass techniques, and concrete recommendations for remediation. In iOS we find a powerful and compelling set of security and privacy controls, backed and empowered by strong encryption, and yet a critical lack in coverage due to under-utilization of these tools leading to serious privacy and security concerns. In Android we find strong protections emerging in the very latest flagship devices, but simultaneously fragmented and inconsistent security and privacy controls, not least due to disconnects between Google and Android phone manufacturers, the deeply lagging rate of Android updates reaching devices, and various software architectural considerations. We also find, in both platforms, exacerbating factors due to increased synchronization of data with cloud services.
>
> The markets for exploits and forensic software tools which target these platforms are alive and well. We aggregate and analyze public records, documentation, articles, and blog postings to categorize and discuss unauthorized bypass of security features by hackers and law enforcement alike. Motivated by an increasing number of cases since Apple v. FBI in 2016, we analyze the concrete impact of forensic tools, and the privacy risks involved in unchecked seizure and search. Then, we provide in-depth analysis of the data potentially accessed via common law enforcement methodologies from both mobile devices and accompanying cloud services.
>
> Our fact-gathering and analysis allow us to make a number recommendations for improving data security on these devices. In both iOS and Android we propose concrete improvements which mitigate or entirely address many concerns we raise, and provide ideation towards resolving the remainder. The mitigations we propose can be largely summarized as the increased coverage of sensitive data via strong encryption, but we detail various challenges and approaches towards this goal and others.
>
> It is our hope that this work stimulates mobile device development and research towards increased security and privacy, promotes understanding as a unique reference of information, and acts as an evidence-based argument for the importance of reliable encryption to privacy, which we believe is both a human right and integral to a functioning democracy.

The detailed analysis PDF is 119 pages. And for anyone who is interested in a much greater level

of detail it is a goldmine reference. But for this podcast, here's a of the team's findings and conclusions for each of the platforms:

In Apple iOS we found a powerful and compelling set of security and privacy controls, backed and empowered by strong encryption. However, we also found a critical lack in coverage due to under-utilization of these tools. More concretely:

- Limited benefit of encryption for powered-on devices. We observed that a surprising amount of sensitive data maintained by built-in applications is protected using a weak "available after first unlock" (AFU) protection class, which does not evict decryption keys from memory when the phone is locked. The impact is that the vast majority of sensitive user data from Apple's built-in applications can be accessed from a phone that is captured and logically exploited while it is in a powered-on (but locked) state.

- Weaknesses of cloud backup and services. Use of Apple iCloud (unsurprisingly) transmits an abundance of user data to Apple's servers, in a form that can be accessed remotely by criminals who gain unauthorized access to a user's cloud account, as well as authorized law enforcement agencies with subpoena power. More surprisingly, we identify several counter-intuitive features of iCloud that increase the vulnerability of this system.

- Evidence of past hardware (SEP) compromise. iOS devices place strict limits on passcode guessing attacks through the assistance of a dedicated processor known as the Secure Enclave processor (SEP). We examined the public investigative record to review evidence that strongly indicates that as of 2018, passcode guessing attacks were feasible on SEP-enabled iPhones using a tool called GrayKey. To our knowledge, this most likely indicates that a software bypass of the SEP was available in-the-wild during this timeframe.

- Limitations of "end-to-end encrypted" cloud services. Several Apple cloud services advertise "end-to-end" encryption in which only the user (with knowledge of a password or passcode) can access cloud-stored data. We find that the end-to-end confidentiality of some encrypted services is undermined when used in tandem with the iCloud backup service. More critically, we observe that Apple's documentation and user settings blur the distinction between "encrypted" (such that Apple has access) and "end-to-end encrypted" in a manner that makes it difficult to understand which data is available to Apple. Finally, we observe a fundamental weakness in the system: Apple can easily cause user data to be re-provisioned to a new (and possibly compromised) HSM simply by presenting a single dialog on a user's phone. We discuss techniques for mitigating this vulnerability.

In Android we found strong protections emerging in the very latest flagship devices, but simultaneously fragmented and inconsistent security and privacy controls, not least due to disconnects between Google and Android phone manufacturers, the deeply lagging rate of Android updates reaching devices, and various software architectural considerations. More specifically:

- Limited benefit of encryption for powered-on devices. Like Apple iOS, Google Android provides encryption for files and data stored on disk. However, Android's encryption mechanisms provide fewer gradations of protection. In particular, Android provides no equivalent of Apple's Complete Protection (CP) encryption

class, which evicts decryption keys from memory shortly after the phone is locked. As a consequence, Android decryption keys remain in memory at all times after "first unlock," and user data is potentially vulnerable to forensic capture.

- De-prioritization of end-to-end encrypted backup. Android incorporates an end-to-end encrypted backup service based on physical hardware devices stored on Google's datacenters. Unfortunately, the end-to-end encrypted backup service must be opted-in to by app developers, and is paralleled by the opt-out Android Auto-Backup, which provides encryption keys to Google servers.

- Large attack surface. Android is the composition of systems developed by various organizations and companies. Because the development of these components is not centralized, cohesively integrating security for all of Android would require significant coordination, and in many cases such efforts are lacking or nonexistent.

- Limited use of end-to-end encryption. End-to-end encryption for messages in Android is only provided by default in third-party messaging applications. Many native Android applications do not provide end-to-end encryption: the exceptions being Google Duo and more recently, the Android Messages application.

- Availability of data in services. Android has deep integration with Google services, such as Drive, Gmail, and Photos. Android phones that utilize these services (the large majority of them) send data to Google, which stores the data under keys it controls - effectively an extension of the lack of end-to-end encryption beyond just messaging services. These services accumulate rich sets of information on users that can be exfiltrated either by knowledgeable criminals (via system compromise) or by law enforcement (via subpoena power).

We also found, in both iOS and Android, exacerbating factors due to increased synchronization of data with cloud services.

And the team's conclusions:

Privacy is critical to a free and open society. For one, people behave differently when they know they might be surveilled , and so privacy-preserving communication channels can alone enable truly democratic discussion of ideas. As more of our data is gathered and our interactions occur on (particularly mobile) devices, these considerations become more important. Worse, harms from violations of privacy are concentrated among some of the most vulnerable, chronically disenfranchised populations in our societies such as the Uighur population in China and Indigenous protesters in the United States .

Smartphones have the potential to fundamentally change the balance of privacy in our lives. Not only do they contain our daily schedules and locations, they store and deliver our communications, rich media content documenting our activities, and are a gateway to the Internet. Our devices also contain information about our families and peers, meaning that compromise of their privacy affects our entire interpersonal network. With the rapid advances of data science, machine learning, and the industry of data aggregation, the potential privacy loss due to phone compromise is difficult to overstate. When we fail to protect our data, we are making privacy decisions not only for ourselves but on behalf of anyone with whom we communicate and interact.
The questions we raise in this work are primarily technical. But they stand for a larger

question: how do we improve privacy for users of mobile devices? In this work we demonstrate significant limitations in existing systems, and show the existence of bypasses to security controls which protect our. We also present ideas which we hope can be adopted, extended, and improved towards the overall goal of improving privacy. Policy and legislative solutions are also very relevant toward this end; we leave this analysis for experts in the respective fields , but to summarize their work, meaningful opportunities for change include limiting law enforcement rights to search devices, requiring robust data sealing and deletion standards, and increasing transparency through audits and logging.

Many of these limitations are centered around what data is encrypted when, and where encryption keys are stored. Encryption, unlike any operating systems security control, lacks the complex state space which contributes to vulnerabilities in software. Whereas the operating system must contend with and manage various user and system contexts, and correctly provide access control and handling from even potentially malicious sources, encryption can be summarized in brief: is the data encrypted using a strong cipher, and where are the keys? We find that while much data on iOS and Android is stored encrypted, the keys are often available in memory. This creates an opportunity for a compromised OS kernel to exfiltrate data as we see in various forensic tools and bypasses. Further, in Android we find that many widely-used but outdated versions of Android offer even more limited coverage of encryption, up to as weak as only encrypting data when the device is off. While modern versions offer strong and more granular file-based encryption, older models are relegated to disk encryption; disk encryption is wholly unprepared for the stronger adversaries we consider in our threat model, where running devices may be seized at any time. In the cloud, both platforms extensively store user data on behalf of devices, and while there are options for end-to-end encrypted content such as app developer opt-in backups on Android and certain data categories on iOS, this coverage is limited due to design decisions by Apple and Google.

Secure hardware offers compelling security benefits on both mobile platforms, particularly by giving the devices a place to store encryption keys without risking their immediate compromise. It is these components, whether the Secure Enclave on iOS or TrustZone on Android, which allow mobile devices to contend in our stronger threat model, far beyond what for example most laptop computers could hope to. Secure hardware is the only reliable method of storing encryption keys on-device and protecting them that we find in the industry, modulo potential bypasses of Secure Enclave technology (e.g. by GrayKey) and vulnerabilities in TrustZone.

Going beyond the current state of the art, there are exciting possibilities to create novel methods for increasing the coverage of encryption without limiting performance. This is a deeply technical challenge which may draw in research from cryptography, systems security, and even machine learning in the example of creating prediction systems for limiting available decryption keys. There are also compelling questions about update distribution and synchronization to be answered. Both platforms could benefit greatly from the work of secure multi-party computation researchers in developing privacy-preserving replacements for sensitive but desirable cloud services such as backup, data aggregation, and predictive systems. Additionally, extensive usable security research and other sociological studies can answer questions about how people expect security and privacy controls to work, and compare any gaps with how they are truly implemented. User studies are difficult and expensive, but also have great potential to aid development research by informing them with a human perspective. Follow-up analysis of the effectiveness of security awareness and education works are also critical in this regard.

Privacy lies at the crossroads of technology and policy, of academic and engineering interest,

and of producer and consumer effort. Effective privacy controls can be mandated through legislative efforts as well as through technical design considerations . Thus there is much opportunity for improvement, but these intersections complicate solutions due to differences in understanding, language, and motivations amongst policymakers and technology designers. The challenges we highlight in this work, then, can largely be solved through mutually informed efforts in the policy and technical domains. We urge researchers and engineers to collaborate with policymakers, advocates, and experts in these efforts. Towards improved usage norms, towards efficient secure protocols to enable sensitive use-cases, and towards mobile device privacy by default for all people.

So, nothing hugely surprising. Mostly a confirmation of what we know and/or have assumed from previous experience with both platforms. But it's useful to have it all pulled together into a single coherent and comparative report. And while there is no much that individuals can do to address the various shortcomings of their chosen platform, it does give Apple and Google some issues to ponder... as does, hopefully, this podcast every week. :)