Security Now! #717 - 06-04-19
The Nansh0u Campaign

This week on Security Now!

This week we check-in on the BlueKeep RDP vulnerability, we look at the planned shutdown of one of the, if not the, most "successful" (if one can call it that) affiliate-based ransomware systems, we update on the anti-Robocalling problem, and then look at the recent announcements by the Russian and Chinese militaries about their plans to move away from the Microsoft Windows OS. We also look at Apple's announcement yesterday of their forthcoming "Sign in with Apple" service and touch on the state-of-SQRL. We then share a bit of fun feedback from a listeners and finish by examining the interesting details behind a significant old-school persistent campaign -- the Nansh0u campaign -- apparent sourced from China, which has successfully compromised many tens of thousands of servers exposed to the Internet.

Wormpocalypse?? ... methinks not.

Courtesy, [The Hacker News]

Security News

Last Thursday, Microsoft: "A Reminder to Update Your Systems to Prevent a Worm"
https://blogs.technet.microsoft.com/msrc/2019/05/30/a-reminder-to-update-your-systems-to-prevent-a-worm/

On May 14, Microsoft released fixes for a critical Remote Code Execution vulnerability, CVE-2019-0708, in Remote Desktop Services – formerly known as Terminal Services – that affects some older versions of Windows. In our previous blog post on this topic we warned that the vulnerability is 'wormable', and that future malware that exploits this vulnerability could propagate from vulnerable computer to vulnerable computer in a similar way as the WannaCry malware spread across the globe in 2017.

Microsoft is confident that an exploit exists for this vulnerability, and if recent reports are accurate, nearly one million computers connected directly to the internet are still vulnerable to CVE-2019-0708. Many more within corporate networks may also be vulnerable. It only takes one vulnerable computer connected to the internet to provide a potential gateway into these corporate networks, where advanced malware could spread, infecting computers across the enterprise. This scenario could be even worse for those who have not kept their internal systems updated with the latest fixes, as any future malware may also attempt further exploitation of vulnerabilities that have already been fixed.

It's been only two weeks since the fix was released and there has been no sign of a worm yet. This does not mean that we're out of the woods. If we look at the events leading up to the start of the WannaCry attacks, they serve to inform the risks of not applying fixes for this vulnerability in a timely manner.

Our recommendation remains the same. We strongly advise that all affected systems should be updated as soon as possible.

It is possible that we won't see this vulnerability incorporated into malware.

But that's not the way to bet.

Then to remind us, Microsoft provided a timeline of the "EternalBlue" exploits...

Almost two months passed between the release of fixes for the EternalBlue vulnerability and when ransomware attacks began. Despite having nearly 60 days to patch their systems, many customers had not. A significant number of these customers were infected by the ransomware.

We had fun last week with the end of the Internet. But I also think there's a very different way to think about this. Somehow, this doesn't really feel to me like a pending worm. One uses a worm for its multiplying effect, when it's necessary to brute-force attack the possibly weak credentials or an authentication weakness which is believed to exist in many, many of which may be unknown. A worm is therefore a good way to collectively pound on the Internet as a whole to break into unknown targets that are resisting. That way you can get newly exploited machines to chime-in on the attack of other vulnerable machines which haven't yet fallen.

But that's not what we have here. Here, there is no need to pound, and the vulnerable machines are not unknown and needing to be pounded upon to be found. IPv4 scanning is now widely available and the code to remotely locate and determine RDP vulnerability is now widely available.

So I really don't see that a worm makes sense. Sure, it IS clearly "wormable", but that wouldn't be optimal. What WOULD be optimal would be for a fast-acting attacker to get their cryptocurrency mining code into as many of those million machines as possible and to then close the door behind them to keep out the competition.

What I would expect to observe, rather than the end of the Internet, or any "Wormpocalypse" would be that, if another RDP scan was performed two or three months from now, it would reveal a great many fewer vulnerable machines... not because they were updated or gone, but because they were now occupied by miners pooling their compute resources.

GandCrab Ransomware Shutting Down After Claiming to Earn $2.5 Billion

https://www.bleepingcomputer.com/news/security/gandcrab-ransomware-shutting-down-after-claiming-to-earn-25-billion/

As we noted in our picture of the week last week, which depicted the timeline of the explosion in ransomware, there are truly too many strains and variations to track. So I don't normally mention the comings and goings of this or that Ransomware of the moment.

However, I have mentioned one of the dominant strains of ransomware, "GandCrab", previously. What’s interesting is that GandCrab was operated as a Ransomware-as-a-Service (RaaS) with an affiliate program. The GandCrab RaaS is an online portal where distributors sign up and pay to get access to custom builds of the GandCrab ransomware, which they then distribute via eMail spam, exploit kits, Remote Desktop Protocol, or whatever means. When an infected user pays a ransom demand, the original GandCrab authors earn a commission, while the rest of the money goes to the person who distributed the ransomware.

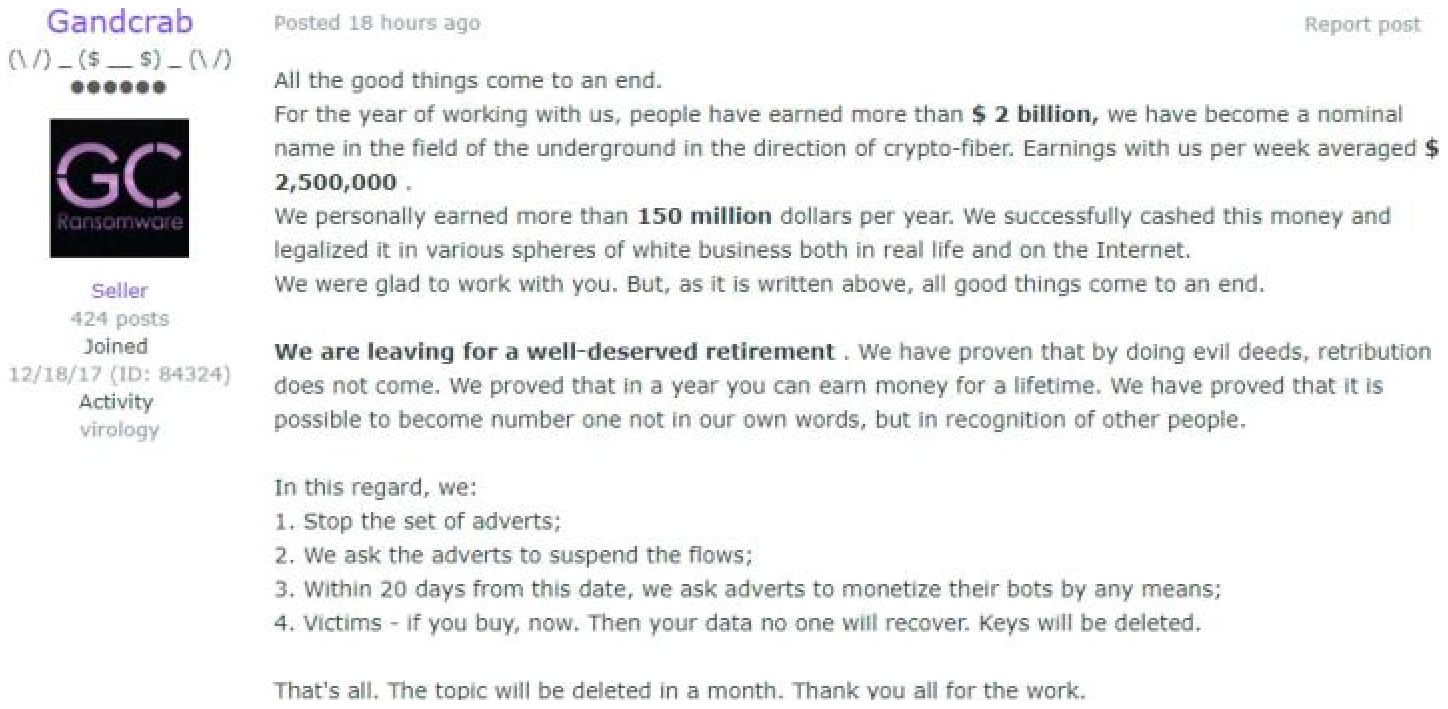
You can just imagine the pitch...

“Yes! You too can form your own money-making, highly-profitable ransomware franchise, encrypting the contents of unwitting strangers’ hard drives and extorting many hundreds of dollars from each of them at no cost to yourself. For just a relatively small piece of the action, we’ll manage all of the messy little details for you. You just go out there, find fresh new victims, and arrange to get them to run the ransom malware we’ll provide to you at no additional charge. We’ll also keep it updated with all of the latest cutting-edge A/V-avoiding tricks & technologies. We’ll even go so far as to purchase bright shiny new code-signing certificates from Comodo -- still, against all reason, one of the most trusted names in Certificate Authorities. That way, the malware you distribute won’t raise any questions when your trusting victims press “Go!” What could be easier? Burdened by the weight of massive student loans? You don’t need Bernie Sanders! Retire that college debt from the comfort of your own parent’s basement!... and show them just what an enterprising little cuss you can be.”

Ah, yes… those were the days. But now, so far as GandCrab is concerned, that dream, which had apparently become a reality for many, will be coming to an end at the end of this month. We're talking about this today because last Friday, May 31st, GandCrab's operators announced their intention to shut down their illicit operation. This stated intention was accompanied by a boast which many who closely follow the machinations of this underworld find doubtful... which was that during the 18 months of its life (the GandCrab portal went live in January of 2018), that GandCrab had netted a total of $2 billion in ransoms.

Now, the GandCrab operators have instructed their affiliates to cease new infections, push existing infected users to pay up and receive decryption keys by the end of the month... after which there will be no way for their data to be recovered.

Two security researchers, Damian and David Montenegro have been tracking the exploits of GandCrab on the underground hacking and malware forum Exploit.in. In images they provided to BleepingComputer, the GandCrab operators state that they have generated more than $2 billion in total ransom payments, with average weekly payments of $2.5 million dollars. They go on to say they have personally earned more than $150 million per year, which they have cashed out and invested in legal business entities.



All the good things come to an end.
For the year of working with us, people have earned more than **$ 2 billion,** we have become a nominal name in the field of the underground in the direction of crypto-fiber. Earnings with us per week averaged **$ 2,500,000** .
We personally earned more than **150 million** dollars per year. We successfully cashed this money and legalized it in various spheres of white business both in real life and on the Internet.
We were glad to work with you. But, as it is written above, all good things come to an end.

We are leaving for a well-deserved retirement . We have proven that by doing evil deeds, retribution does not come. We proved that in a year you can earn money for a lifetime. We have proved that it is possible to become number one not in our own words, but in recognition of other people.

In this regard, we:
1. Stop the set of adverts;
2. We ask the adverts to suspend the flows;
3. Within 20 days from this date, we ask adverts to monetize their bots by any means;
4. Victims - if you buy, now. Then your data no one will recover. Keys will be deleted.

That's all. The topic will be deleted in a month. Thank you all for the work.

So GandCrab has said they have stopped promoting their ransomware, they have asked their affiliates to stop distributing the ransomware within 20 days, and asked their forum storefront "topic" to be deleted at the end of the month.

BleepingComputer noted in their coverage of this:

They have also told victims to pay for needed decryption now as their keys will be deleted at the end of the month. This is could be a last money grab and we hope that the GandCrab devs will follow other large ransomware operations and release the keys when shutting down.

BleepingComputer has reached out to the developers and asked them to do so.

Historically, BleepingComputer has seen large-scale ransomware operations fill the void left when another ransomware shuts down. It would not be surprising to see another operation spring up in the near future.

Incremental forward progress on the RoboCalling front
We talked previously about the SHAKEN and STIR telecommunications protocols which are in the process of being gradually deployed by all of the major telecommunications carriers. The event of our previous discussion was the first inter-carrier test of the technology between AT&T and Comcast. Until then all work had been intra-carrier. But it's only when calls can be verified from end to end that the system provides meaningful anti-spoofing protetion, thus is needs to be universal and cross-carrier.

As we noted at the time, SHAKEN & STIR apply certificate-based callerID signing to strongly prevent callerID spoofing which is rampant today since there is absolutely zero technology in place to prevent it.

The reason this comes up again, today, is that the US Senate just passed on a vote of 97 to 1, a bill known as TRACED -- Telephone Robocall Abuse Criminal Enforcement and Deterrence Act --which will now go to the House of Representatives, where it is expected to pass just as handily, then to the President's desk for signing to law.

When the bill becomes a law, it will empower the Federal Communications Commission (our FCC) to impose significant fines – up to $10,000 per call – for illegal robocalls. The legislation would also increase the statute of limitations for bringing such cases, thereby giving FCC regulators more time to track down offenders.

The act also creates an interagency task force to address the problem, and pushes the Us's major carriers such as AT&T and Verizon to continue moving forward with the deployment of the SHAKEN & STIR call authentication systems.

So this is all looking good. This most recent bill (there have been 13 preceding) is the toughest yet and it has the backing of all 50 state attorneys general, 35 of whom told the FCC last October that they were pulling their hair out over the enormous problem which was beyond anything their states' law enforcement agencies could handle.

Earlier this year, in February, the FCC Chairman Ajit Pai reiterated his call for a robust caller ID authentication system to be implemented this year. Earlier this month, Pai announced a new FCC initiative to fight illegal robocalls that would assure carriers that they're able to automatically register customers for call-blocking service. At this point, customers have to do it themselves.

I skipped over the news about three weeks ago because it didn't quite make the grade, but it fits nicely in with this update: On Wednesday, May 15th, the FCC also announced a new measure that would grant mobile phone carriers new authority to block the robocalls. The new rule would make it easier for our mobile carriers, like AT&T, Verizon, and T-Mobile, to automatically register their customers for call-blocking technology rather than, as it is now, requiring consumers to opt-in on their own. It would also allow customers to block calls coming from phone numbers that are not on their contacts list -- which would be such a perfect filter. The FCC commissioners will be voting on this one this coming Thursday the 6th.

In this instance, Ajit Pai said: “Allowing call blocking by default could be a big benefit for consumers who are sick and tired of robocalls. By making it clear that such call blocking is allowed, the FCC will give voice service providers the legal certainty they need to block unwanted calls from the outset so that consumers never have to receive them.”

So... as we see, these things never happen quickly. And it had to get virtually intolerable before moving to fix it began... but is does appear that before long this will be a memory.

Chinese and Russian militaries are moving away from Windows

For their military systems, Russian authorities have continued to move toward implementing their plan to replace Windows with a locally-developed operating system named Astra Linux.

Wikipedia has this to say about Astra Linux: Astra Linux is a Russian Linux-based computer operating system developed to meet the needs of the Russian army, other armed forces and intelligence agencies. It provides data protection up to the level of "top secret" in Russian classified information grade. It has been officially certified by Russian Defense Ministry, Federal Service for Technical and Export Control and Federal Security Service.

So, last month, the Russian Federal Service for Technical and Export Control (FSTEC) granted Astra Linux the security clearance of "special importance," which means the OS can now be used to handle Russian government information of the highest degree of secrecy. Before this, the Russian government had been using special versions of Windows that had been modified, checked, and approved for use by the FSB.

Now, the Russian military will begin their move to Astra Linux, a Debian derivative which has been in the works for eleven years, by the Russian company RusBITech. RusBITech initially developed the OS for use in the Russian private market, but the company also expanded into the local government sector, where it became very popular with military contractors.

Several years ago, the OS received certifications to handle Russian government information at the two levels below "special importance", which are "secret" and "top secret." Since then, Astra Linux has slowly made its way into government agencies and is currently in use at the Russian National Center for Defence Control, among various other government and military agencies.

A year and a half ago at the start of 2018, the Russian Ministry of Defence announced their intentions to transfer military systems from Windows to Astra Linux, citing fears that Microsoft's closed-source approach might hide Windows backdoors that could be abused by US intelligence to spy on Russian government operations. And, as our listeners know, I've been shaking my head in amazement for years that Russia could still be using Windows for ANYTHING!

So, for this past 18 months, RusBITech has been pursuing the Russian government's certification process to obtain a "special importance" classification for Astra Linux -- which finally happened just this past April 17. In addition to the FSTEC certification, Astra Linux also received certificates of conformity from the FSB (we know who they are... Russia's top intelligence agency) as well as the Ministry of Defense, which opens the door to full adoption by all of Russia's top military and intelligence agencies.

Meanwhile, over in China… Chinese military also plans to replace Windows due to fears of US hacking, but in their case they'll be moving to a custom OS, not Linux.

According to reports, officials in Beijing have decided to develop a custom Chinese OS to replace Windows on computers used by the Chinese military. The report did not come by way of official Chinese channels, but by way of a Canada-based military magazine Kanwa Asian Defence.

As the reporting goes, due to the Edward Snowden, Shadow Brokers, and Vault7 leaks, Beijing officials are quite well aware of the US' hefty arsenal of hacking tools, able to target anything from smart TVs to Linux servers, and from routers to common desktop operating systems, such as Windows and Mac.

Since these revelations have shown that the US can hack into almost anything, the reporting has characterized the Chinese government's plan as that adoption of a "security by obscurity" approach by running a custom OS that will make it harder for foreign threat actors -- mainly the US -- to spy on Chinese military operations.

I would characterize the move differently, and I think it makes a lot of sense, though it's going to be a heavy lift to create a desktop-class OS from scratch. I don't see it as security through obscurity but rather as a move away from a monoculture OS environment. Monocultures are inherently fragile. Increased heterogeneity among OSes hugely reduces the threat from both accidental mishap and deliberate attack. With someone like SandboxEscaper spewing out privilege elevation exploits daily, lord only knows how many ways into Windows our own US intelligence services have discovered and stockpiled. As I've said before... I'm utterly amazed that Windows has been allowed to exist inside rival foreign governments. It must be that any Windows machines being used in sensitive environments are well-isolated and NEVER experience and encrypted real-time communications with the Internet.

Anyway... the task of developing the new OS and replacing Windows goes to a new "Internet Security Information Leadership Group," first reported by the Epoch Times, citing the May issue of the Kanwa Asian Defence magazine. Per the magazine, this new group answers directly to the Central Committee of the Chinese Communist Party (CCP) and is separate from the rest of the military and intelligence apparatus.

It was be so much fun to be assigned the task of creating a new OS from scratch. It would also be so tempting to borrow from the existing extensive open source code base. But there no way to do that without the possibility of inadvertently carrying over unsuspected vulnerabilities.

At the same time, let's recall how long it took us to get something as relatively simple and straightforward as a TCP/IP protocol stack that wasn't an utter disaster of edge cases and security flaws. Think about the man-centuries of effort that has gone into our web browsers and their rendering engines and scripting interpreters. The more I think about it, the less reasonable -- or possible -- it sounds to start over, now, from scratch. It would really be impossible... unless you want DOS.

Sign in with Apple

During yesterday's Apple WWDC 2019 kickoff, Apple announced their launch of a privacy-respecting "Sign-In with Apple" OAuth redirection service similar to the very familiar "Sign-In with Google" and "Sign-In with Facebook".

They also beat all other OAuth providers to the punch by adding a feature to their forthcoming OAuth redirection service which I'll call: "managed random eMail address provisioning". This is a brilliant idea, and it's a reason for using Apple's OAuth service over others, even if one isn't concerned about OAuth's tracking.

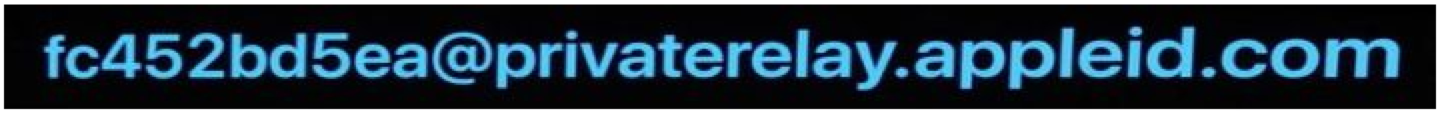
So first let's back up a bit...

As we know, OAuth is a convenience, but it also comes at the clear cost of explicit and high-reliability tracking. When our browsers are redirected though that chosen 3rd-party authentication provider, that 3rd-party knows who we are -- since we have an account with them -- and they know to which site we are signing in, since they are negotiating our identity with that site, on our behalf, behind the scenes. And all of this uses 1st-party session cookies.

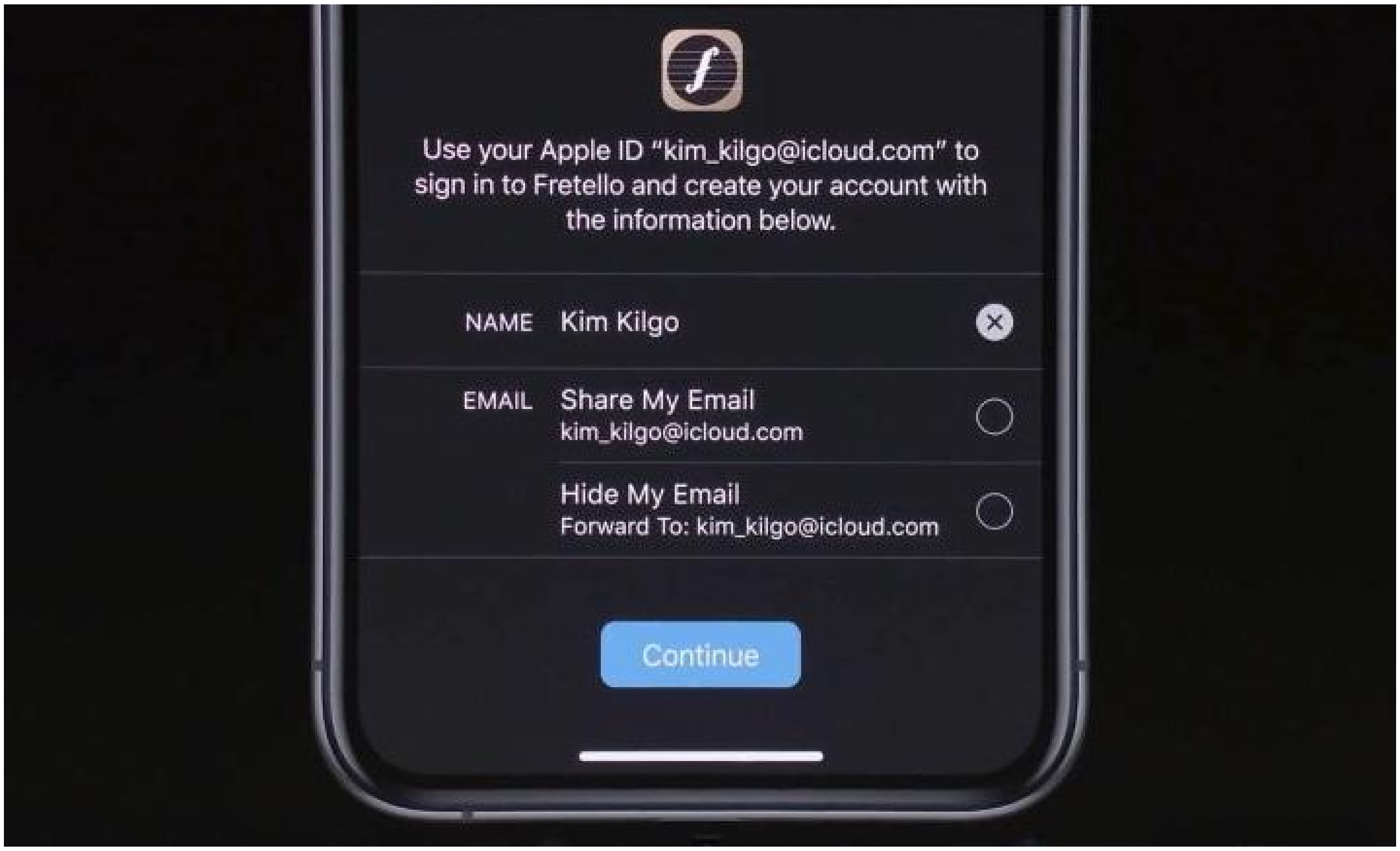
So we know that there's just no way that Facebook and Google, both having strong financial interests in compiling activity profiles about everyone, are not actively leveraging that very specific and clear who-you-are and where-you're-going information. As we know, traditional browser tracking is inherently soft and fuzzy and requires some degree of behavioral inference. Not so with explicit "sign-in with..." OAuth. So on the tracking privacy front this was clearly another poke in the eye by Apple to all of the other OAuth trackers.

The problem is that, for the experience to be seamless, our browsers need to be maintaining an active session-state with the OAuth provider, so that our browsers can bounce though them transparently. But I have no such relationship with Facebook. And although I'm currently an avid Apple device consumer, I don't currently have such a relationship with Apple, because Apple is not, otherwise, a provider of online services for me. But I do have a relationship with Google. Every one of my various browsers maintain persistent sessions with Google... because Google already offers many online services which I use. And while I dislike the idea of being tracked, it's not really something I worry about. I strongly HATE the idea from a technology standpoint of it NOT being under my control, if I did care. That seems really wrong. And I certainly understand that many people DO care.

Remember, though, that as part of the OAuth transaction, the site being signed into can request data from the site you're signing in =with= ... such as your username and your eMail address. Since this happens behind the scenes, users may not even be aware of it. Or if they are show that this will be happening, they're typically not given the option to not have that information provided. So, what may mean a great deal to many people who deeply trust Apple, (and in a moment we'll see why "deep trust" is required), is that rather than providing the service that I'm being signed into my =real= Apple eMail account, (which would *really* annoy me because that's a unique account I use only for Apple), Apple will be offering the option to proactively mask their user's actual eMail by providing a randomly chosen eMail "relay address"...


fc452bd5ea@privaterelay.appleid.com

I called this "managed random eMail address provisioning" since, on stage yesterday, Craig Federighi indicated that Apple users would have some means for curating and removing the relay addresses from any sites from which they no longer wish to receive eMail. This DOES place Apple in the enviable position of receiving -- and viewing if they wished -- and then forwarding ALL of the eMail being received.


Use your Apple ID “kim_kilgo@icloud.com” to sign in to Fretello and create your account with the information below.

NAME Kim Kilgo

EMAIL Share My Email
kim_kilgo@icloud.com

Hide My Email
Forward To: kim_kilgo@icloud.com

That's probably not a big deal since Google already receives and scans all of the eMail pouring through their servers. And Apple certainly promotes their enforcement of our privacy rights much more strongly than either Facebook or Google. But it is something to keep in mind. For me, my Google eMail is already my designated "junk" eMail bucket since its anti-spam filtering is so good. And today my Apple eMail account has remained utterly pure since I have never used it for anything else. So I think I'd rather be using "Sign in with Google" and have that eMail address disseminated than to start polluting the eMail account I share with Apple.

But, interestingly, one reason to think that it WILL become prevalent, at least from within apps over which Apple has control (like those for iOS) is that offering it will NOT be optional. Down in the fine print at the end of Apple's new App Store Review Guidelines Apple states: "Sign In with Apple will be available for beta testing this summer. It will be required as an option for users in apps that support third-party sign-in when it is commercially available later this year."

So any apps which offer OAuth third-party logins like Facebook, Google or whatever, MUST also offer Apple's own sign in service as well. User's won't need to choose Apple, but the option will need to be presented.

SQRL:

There's no way I can talk about all of the problems with OAuth and what a mess all of this is, without noting that there IS a working system which is now in place, that has none -- not a one -- of these issues and problems… and that, of course, is the system that has been my primary focus for the past five years. There cannot be any tracking nor any form of information disclosure because SQRL was designed for all of these reasons to be a fully functional, 2-party authentication system -- between just you and the site you are wishing to establish a relationship with. SQRL instantly, uniquely and pseudonymously identifies its users to new websites and revisted websites without revealing ANY information of any kind whatsoever. The site receives only a long unique string that only that SQRL user will present to only that website.

I know that I've been talking about this for years, but it's all finally finished. The SQRL forum site now has 1,275 registered and active members, all of whom are using SQRL clients, daily, with Windows, Linux, iOS, Android, Firefox, Chrome and Edge to login without using or needing to supply usernames, passwords or eMail addresses.

I am very nearly finished with the first section of the final SQRL explainer and protocol specification document. I had hoped to have the first section ready to announce and offer by today's podcast, but I missed that mark. I still need to complete the Identity Replacement and SQRL Service Provider API overviews. But it **will** be ready by next week and I'll have a link for everyone to download the PDF. And by then I will also already be working on the second half, which is the fully articulated SQRL protocol specification. As with the first half of the document, I'm just documenting what has already been done, what already exists, has been proven, and is in active use.

My own SQRL reference client for Windows appears to be completely finished. It's been several weeks since it's last update and release. So I expect to make it widely public, shortly.

In other words… everything is finally just about ready.

Closing the Loop Feedback *(regarding "RDP: Really Do Patch")*

ChristopheVanlancker / @Carroarmato0
Dear @SGgrc , you mispronounced Ransomware Deployment Protocol in the last episode :)

SpinRite

Hello Mr. Gibson,

I've been a listener on Security Now for a few years now and always enjoy seeing a new episode show up on my phone on Wednesday mornings.

After listening to episode 702 and after hearing (not even seeing!) you and Leo talk about the PiDP-11, I knew I had to have one. Beyond building it and getting it running, my main interest was using the front panel switches to do something interesting, such as toggling in the bootstrap loader as you mentioned doing on other PDP machines during the episode. I wasn't able to find any instructions or examples of people doing this on the PiDP-11, so I decided to figure it out for myself.

After finding the bootstrap code for the RT-11 operating system, I published a short Youtube video today discussing how to configure the PiDP-11 software and toggle the correct switches in order to boot the RT-11 OS, which is included in the PiDP-11 software distribution. The Google Group discussion thread is: https://groups.google.com/forum/#!msg/pidp-11/1CVuv7UZpFc/q2ZWP98MCAAJ and the video itself is at: https://www.youtube.com/watch?v=31vrju0BMQc Both you and Leo might be interested in trying this as it is fairly quick and entertaining to toggle switches and see something useful occur.

Obligatory SpinRite Story:

This is a fairly mundane success story, but I have a NetBSD server which does a nightly backup to an 8 year old 2TB hard drive. Since this is a nightly task that has run for years, I have been able to watch the hard drive transfer rate steadily decreasing over the past few months until the point where it was obvious that something was wrong, even though the SMART data that I had also been collecting didn't seem unusual or to be deteriorating.

I had been meaning to run SpinRite on the drive but didn't get around to it until last week, and when I tried to halt the system, I discovered that it was running so slowly that several night's worth of backups were still running, the drive was completely unresponsive, and I eventually just had to power down the machine.I ran SpinRite on Level 4 for a little over 24 hours - there were no errors reported and, again, the SMART data seemed ok. I reinstalled the drive into the server and it is now working correctly and performing at its normal speed!

Thanks again for the podcast and SpinRite!

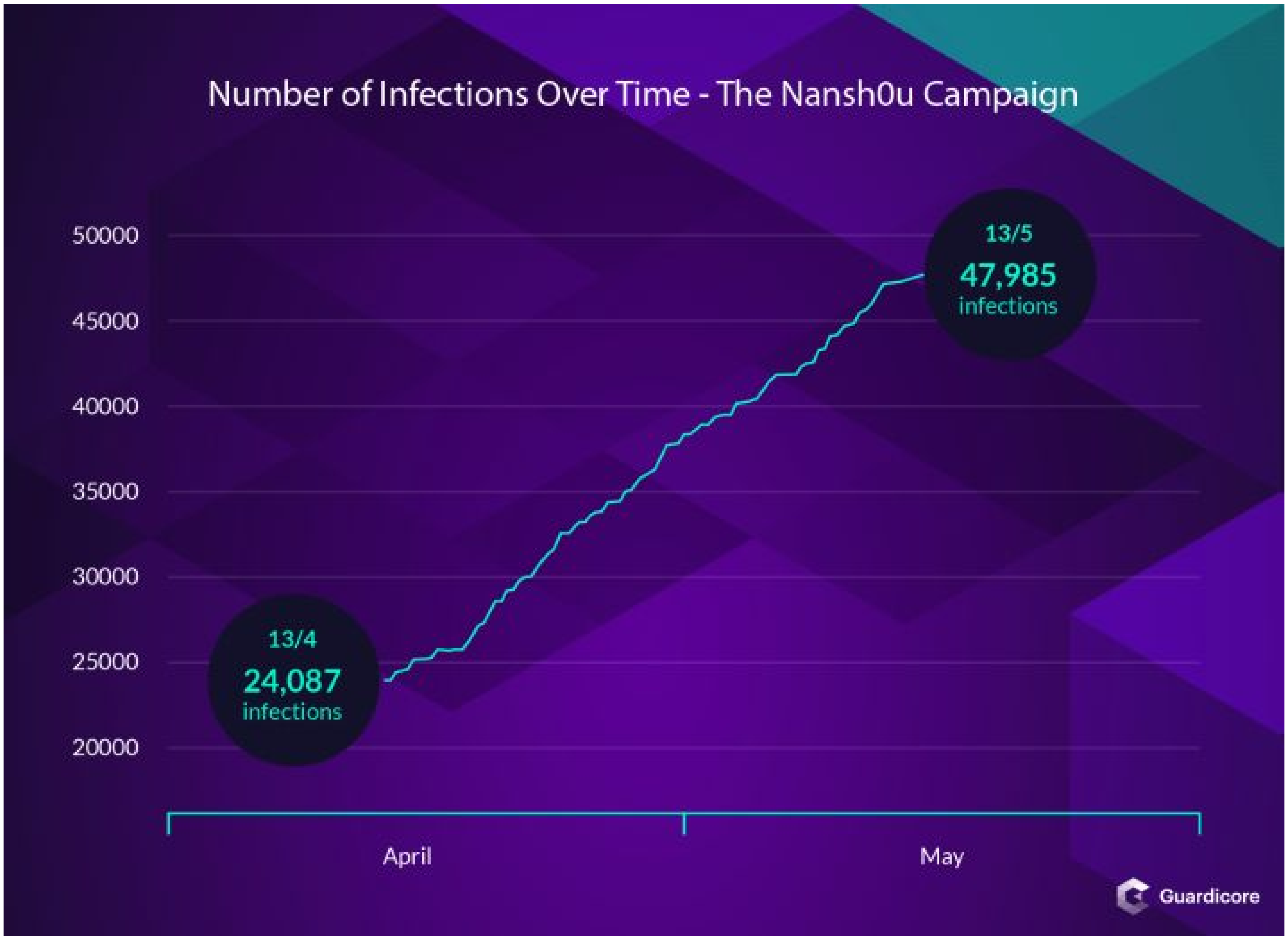
Jeff Thieleke (pronounced: Tell-key...)

The Nansh0u Campaign

During the past two months, the Guardicore Labs team has been closely following a China-based campaign which aimed to infect Windows MS-SQL and PHPMyAdmin servers worldwide. We have taken a deep look into the inner workings of the campaign – the tools in use, the vulnerabilities exploited and the extent of damage caused.

https://github.com/guardicore/labs_campaigns/tree/master/Nansh0u

Breached machines include over 50,000 servers belonging to companies in the healthcare, telecommunications, media and IT sectors. Once compromised, the targeted servers were infected with malicious payloads. These, in turn, dropped a crypto-miner and installed a sophisticated kernel-mode rootkit to prevent the malware from being terminated.



In the beginning of April, three attacks detected in the Guardicore Global Sensor Network (GGSN) caught our attention. All three had source IP addresses originating in South-Africa and hosted by *VolumeDrive* ISP (see IoCs). The incidents shared the same attack process, focusing on the same service and using the same breach method and post-compromise steps.

Looking for more attacks with a similar pattern, we found attacks dating back to February 26, with over seven hundred new victims per day. During our investigation, we found 20 versions of malicious payloads, with new payloads created at least once a week and used immediately after their creation time.

This timeline, combined with a set of five attack servers and six connect-back servers, suggests an established process of continuous development which was well thought of by the attackers. Having access [then] to the attacker's infrastructure, we were able to monitor the relevant file servers and draw insights on the campaign's size and scope. The graph on the page above shows how the number of successfully-infected machines doubled within a timespan of one month.

Each attack started with a series of authentication attempts to a MS-SQL server, eventually leading to a successful login with administrative privileges.Then, a sequence of MS-SQL commands was executed to accomplish the following (numbers in brackets indicate the relevant lines of code):

- Configure server settings to allow a "smooth" and errorless attack flow;
- Create a Visual-Basic script file in *c:\ProgramData\2.vbs*;
- Execute this script and download two files to c:\ProgramData over HTTP; and
- Run the two files in one command-line.

The attacker's servers were all running HFS – HTTP File Server – serving files of different types. One text file included the string *Nansh0u*, on which we based the campaign's name.
The attacker's infrastructure contained all the modules required for a successful, end-to-end attack on MS-SQL servers:

1. Port scanner
2. MS-SQL brute-force tool
3. Remote Code Executor

The port scanner used by the attacker had been known since 2014. Our attacker used it to detect MS-SQL servers by scanning IP addresses and checking whether typical MS-SQL ports were open. The ports examined were: 1433, 2433, 1533, 1143, 9433, 5433.

The results of the scanning were then fed to the brute-force module.

The brute-forcing tool attempts to login to each MS-SQL server using tens of thousands of common credentials. Once authentication succeeds – the server's address, username and password are saved to a file for future use.

The dictionaries of common credentials used by the attacker can be found in the campaign's IoCs repository on Github.

By running the port scanner and the bruteforce tool, the attacker gained a list of breached servers – IP addresses, ports, usernames and passwords. The next step was to login to the victims and infect the machines.

On the HFS (HTTP File Server), in a folder named *chuan* – Chinese for *infect* – we found two interesting components. The first was a script named *Mssql.log*. The script's commands were the exact ones we saw in the incidents that triggered our research.

The second file was an executable named *Usp10.exe*. This program receives a server's address and credentials as well as the contents of *Mssql.log*. Its functionality is straightforward: it logs into the breached server and executes the *Mssql.log* script, proceeding with the attack on the victim side.

Usp10.exe is the attack module responsible for writing and executing the downloader script *(2.vbs)* on the MS-SQL victims. Once the MS-SQL script is done running on the victim machine, the malicious payload is executed.

In all the attacks we have seen as part of this campaign, the attacker executed a command line with two executables:

```
C:\Windows\system32\cmd.exe /c c:\ProgramData\apexp.exe c:\ProgramData\tl.exe
```

This command executes a privilege escalation exploit that runs the malicious payload with *SYSTEM* privileges. In the attacker's arsenal are two versions of the PE exploit, *apexp.exe* and *apexp2012.exe*, and many payload versions.

apexp.exe and apexp2012.exe are two exploits of a known privilege escalation vulnerability (CVE-2014-4113). Passing any program to these executables will run it with SYSTEM privileges.

apexp.exe is known as the Apolmy exploit and it affects both Desktop and Server versions of Windows (XP to 8.1 and Server 2003 to 2012 R2, respectively). This is a weaponized exploit with production-level code. apexp2012.exe, on the other hand, resembles more of a proof-of-concept than an operational exploit, and is designed to work on Windows 8.1. We found the latter available for download on a Chinese forum seemingly used as a hacker-community platform.

While both versions use the same vulnerability, they execute kernel-mode code for different purposes. The Apolmy version copies the SYSTEM process access token to its own process. With that token, the exploiting process runs the payload with full control over the victim machine.

The second version uses a method popularized by Cesar Cerrudo. Here, the exploit adds the SeDebugPrivilege to the token. Using this Windows privilege, the attacking exploit injects code into the winlogon process. The injected code creates a new process which inherits winlogon's SYSTEM privileges, providing equivalent permissions as the prior version.

We collected 20 payload samples from both the attacker's servers and Guardicore Global Sensor Network (GGSN). Each payload is in fact a wrapper and has several functionalities:

- Execute the crypto-currency miner;
- Create persistency by writing registry run-keys;
- Protect the miner process from termination using a kernel-mode rootkit;
- Ensure the miner's continuous execution using a watchdog mechanism.

The payloads spawn one of two processes – dllhot.exe or canlang.exe, depending on the payload – which mine a privacy crypto-currency named TurtleCoin for four different mining pools.

Many of the payloads drop a kernel-mode driver, named randomly and placed in *AppData/Local/Temp*. Its compile time suggests that it had been created in 2016. Nevertheless, most AV engines do not detect the driver file as malicious.

We found that the driver had a digital signature issued by Verisign. The certificate – which is expired – bears the name of a fake Chinese company – *Hangzhou Hootian Network Technology*. Guardicore Labs contacted *Verisign* and provided them with the relevant details, which resulted in the certificate's revocation.

This would have been less awkward, had the driver not been packed and obfuscated. Unlike many other malicious drivers, this driver is protected and obfuscated with VMProtect, a software tool that attempts to frustrate reverse engineers and malware researchers.

The driver is designed to protect processes and prevent the user from terminating them. It creates a device named *SA6482*, allowing processes to communicate with it. The device receives process IDs (PIDs) meant to be protected – in our case, the miner PID.

The driver protects the process by registering a callback on the Process and Thread object types. These callbacks are triggered with every access to the protected process or any of its threads and allows drivers to modify the access rights given in each access attempt.

The driver itself contains additional rootkit functionality such as communicating with physical hardware devices and modifying internal Windows process objects that are unused by this particular malware.

The infection process uses advanced technological capabilities. *apexp.exe*, for instance, takes advantage of a kernel-mode vulnerability to execute code with *SYSTEM* privileges. It was used in a previous Chinese APT by a highly-skilled adversary to target various companies.

Another example is the driver dropped by the different payloads. Obtaining a signed certificate for a packed driver is not at all trivial and requires serious planning and execution. In addition, the driver supports practically every version of Windows from Windows 7 to Windows 10, including beta versions. This exhaustive coverage is not the work of a hacker writing a rootkit for fun.

These advanced elements are opposed to several odd SecOps decisions taken by the attacker.

To begin with, attackers usually do not keep their whole infrastructure on a file server with no activated authentication controls. Logs, victims lists, usernames, binary files – we had them all in a mouse click. In addition, all binary files had their original timestamps; an experienced malware author would have tampered with those to complicate the analysis process.

This campaign was clearly engineered from the phase of IPs scan until the infection of victim machines and mining the crypto-coin. However, various typos and mistakes imply that this was not a thoroughly-tested operation. For example, we found a mismatch in two versions of the *lcn.exe* payload. Both were running the same miner but with swapped command-line arguments, suggesting that the first one was providing the wallet address in an incorrect position.

Last but not least, many of the programs found on the server are written in EPL – Easy Programming Language. EPL is a proprietary, Chinese-based programming language designed for rapid application development. The decision to write a major part of the infrastructure in a relatively esoteric language is unusual.

Advanced vulnerability exploitations are not a common sight in the repertoire of crypto-mining attacks. Similarly, obfuscated kernel code usually does not come bundled in campaigns aiming for altcoin.

It appears that tools, which until recently belonged to nation state-level hackers, are today the property of even common criminals.

We can confidently say that this campaign has been operated by Chinese attackers. We base this hypothesis on the following observations:

- The attacker chose to write their tools with EPL, a Chinese-based programming language.
- Some of the file servers deployed for this campaign are HFSs in Chinese.
- Many log files and binaries on the servers included Chinese strings, such as 结果-去重复 ("duplicates removed") in logs containing breached machines, or 开始 ("start") in the name of the script initiating port scans.

What enables this attack on Windows MS-SQL servers in the first place, is having weak usernames and passwords for authentication. As trivial as it may sound, having strong credentials is the difference between an infected and a clean machine.

If you expose database servers to the internet, you should minimize the danger of any possible compromise. Database vendors typically provide hardening guides; Microsoft's guide for SQL server is pretty easy to implement. In addition, you should separate internet-exposed servers from internal servers as much as possible by segmenting your network and limiting the blast radius of vulnerable devices.

We have released an open source PowerShell script to provide defenders with a simple way to detect infected machines. For more details and instructions check our repository.

The Nansh0u campaign is not another run-of-the-mill mining attack. It serves as a case study for a thoroughly-planned attack scheme which incorporates various modules and progressive tools. This suggests that cyber weapons are becoming readily available, and not only to nation-state organizations. This campaign demonstrates once again that common passwords still comprise the weakest link in today's attack flows. Seeing tens of thousands of servers compromised by a simple brute-force attack, we highly recommend that organizations protect their assets with strong credentials as well as network segmentation solutions.