Security Now! #748 - 01-07-20
Our Malware Lexicon

This week on Security Now!

This first podcast of 2020 we look at a proposed standard for creating machine-readable warrant canaries. We also take a precautionary lesson from a big Xiaomi blunder, exmaine Microsoft's research into brute-forcing RDP, look at the continuing problem at the Point Of Sale, follow-up on Russia's plan to disconnect from the Internet, consider the end of life of Python 2.7, review the top 20 HackerOne bounty payers, warn of some bad new SQLite security vulnerabilities and cover a bit of Sci-Fi, SQRL and SpinRite miscellany. Then we group all malware into a seven-member Lexicon and quickly run through each one.

https://github.com/canarytail/standard

Security News

A proposed standard for making warrant canaries machine readable.
This tidbit popped-up a few weeks ago and I didn't want to skip it:
https://nakedsecurity.sophos.com/2019/12/19/proposed-standard-would-make-warrant-canaries-machine-readable/

The term "Canary" comes from the idea of taking a canary down into a coal mine as a low-tech oxygen and noxious gas monitor. Wiktionary defines a "Canary in a coal mine" as "Something whose sensitivity to adverse conditions makes it a useful early indicator of such conditions; something which warns of the coming of greater danger or trouble by a deterioration in its health or welfare."

Which brings us to "Warrant Canaries"...

The trouble was that government warrants compelling the release of information were typically accompanied by gag orders enjoining the served party from making any disclosures about being in receipt of such a warrant. In other words: "You must provide us with the following information and you cannot tell anyone that we asked and you provided."

When it was passed in 2001, the US Patriot Act enabled authorities to access personal information stored by a service provider about US citizens. It also let them issue gag orders that would prevent the organisation from telling anyone about it. It meant that the government could access an individual's private information without that person knowing.

Privacy respecting companies like ISPs and cloud service providers want their users to know whether the government is asking for this information. The idea of the Warrant Canary was first conceived by a guy named Steve Schear in 2002 after the Patriot Act came into effect. The idea was to provide a passive way of warning people that an organisation holding their data has received a subpoena.

The idea is sheer brilliance: Instead of an organization telling people that it has been served with a subpoena (which is illegal under the Patriot Act), the organisation stops telling them that it hasn't been service with a subpoena.

To do this, the organization displays a public statement online that it only changes if the authorities serve it with a warrant. As long as the statement stays unchanged, individuals know that their information is safe. But if the "nothing to see here, move along" statement changes or disappears, they can infer that all is not well without the organisation explicitly saying so.

The idea is clever, but, until now it hasn't scaled well. That may be about to change. The problem is that these statements have been ad hoc and designed to be read, interpreted and understood by people. This makes them difficult to track and monitor at scale... Which is what a warrant canary standard would solve.

A classic example is what happened with Spider Oak during the summer of 2018. They were forced to try to back everyone away from the ledge, which raise questions about whether they

were being compelled to do that, too. On August 18th, 2018 they posted:
https://medium.com/@SpiderOak/a-transparency-report-is-a-canary-spideroak-62cb235016b1

> Over the weekend there has been chatter on the internet about the change at SpiderOak from a Warrant Canary to a Transparency Report. We understand that some people are concerned that this is a signal that we have in some way been compromised. To be completely clear: Nothing has changed other than the way we report our interactions with the government from the first time we posted a canary in August 2014.
>
> We have received: 0 Search Warrants, 0 Subpoenas, 0 Court Orders, and 0 National Security Letters.
>
> Even better for our customers, we couldn't hand over their data even if we were asked to. The No Knowledge approach that SpiderOak uses means that we don't have the keys to decrypt the data you trust us to store for you.
>
> Thank you for choosing SpiderOak and deciding to trust in cryptography instead of promises.

To combat all of these problems the proposed "CanaryTail" standard surfaced on Github a little over two weeks ago: https://github.com/canarytail/standard

Created by GitHub user "carrotcypher", it was inspired by the work of organisations like the Calyx Institute, a technology non-profit that develops free privacy software, and the now-defunct Canary Watch project from the Electronic Frontier Foundation (EFF), Freedom of the Press Foundation, NYU Law, Calyx and the Berkman Center. Canary Watch listed and tracked warrant canaries. When Canary Watch was shut down, the EFF explained:

> In our time working with Canary Watch we have seen many canaries go away and come back, fail to be updated, or disappear altogether along with the website that was hosting it. Until the gag orders accompanying national security requests are struck down as unconstitutional, there is no way to know for certain whether a canary change is a true indicator. Instead the reader is forced to rely on speculation and circumstantial evidence to decide what the meaning of a missing or changed canary is.

"Canary Tail" hopes to change all of this. As it explains on its Github readme.md page: We seek to resolve those issues through a proper standardized model of generation, administration, and distribution, with variance allowed only inside the boundaries of a defined protocol.

So, instead of some arbitrary language on a website, the warrant canary standard would be a well-defined JSON-formatted text file which is convenient since it's readable by people and machines. The CanaryTail file would include 11 codes with a value of zero (false) or one (true). These codes include WAR for warrants, GAG for gag orders, and TRAP for trap and trace orders, along with another code for subpoenas, all having specific legal meanings for an organisation and its users. A value of '0' adjacent to any of these keys would mean that none of the warnings have been triggered. If the code changes to one, it's cause for concern.

The file also contains some other interesting codes, including DURESS, which indicates that the organisation is being coerced somehow, along with codes indicating that they have been raided. There is also a special code indicating a Seppuku pledge -- as in the ancient Japanese suicide ritual. In this case it's a pledge that an organisation will shut down and destroy all its data if a malicious entity takes control of it.

And because these days you can't having anything good without somehow tying in Bitcoin's blockchain, the proposed standard must be cryptographically signed with a public key, including information about the expiry date. And including a block hash from the bitcoin blockchain to verify the freshness of the digital signature. As another safeguard, it includes a PANICKEY field with another public key. If the file is signed with this key, people can interpret it as a kill switch, causing the warrant canary to fail immediately. That’s useful if an organisation suddenly gets raided and can’t afford to wait until the current warrant canary file expires.

Who knows whether or not this bird will fly. But it's fun to see creating solutions to both old and new problems.

Putting the "Yow!" in Xiaomi…

Recently, a number of users of Xiaomi's smart (or apparently not-so-smart) IP-connected IoT cameras have been receiving and watching other Xiaomi users' video feeds. Whoops.

For whatever reason, what is clearly an unintended bug appears to only affect Xiaomi IP cameras when they are streaming through Google's Nest Hub. The problem first came to light when a Reddit user claimed that his Google Nest Hub was apparently pulling random feeds from other users instead of his own Xiaomi Mijia cameras. He shared some photos showing other people's homes, an older adult sleeping on a chair, and a baby sleeping in its crib... All which appeared on his Nest Hub's screen.

The problem doesn't appear to lie in Google's implementation, but rather in the way Xiaomi implemented the connection between its smart cameras and Nest Hub. In a statementt Google provided to the publication "Android Police", it said that it is aware of the issue and is in contact with Xiaomi to fix the problem, although Xiaomi hadn't yet responded. So, in the meantime, as a precaution to protect its Nest Hub users, Google has temporarily disabled all Xiaomi devices' access to its Nest Hub and Assistant applications. THAT will certainly get Xiaomi's users' attention and, in turn, Xiaomi's!

The exact details surrounding the bug remain unknown, but it doesn't appear to be affecting all Xiaomi cameras which have been integrated with the Google Nest Hub. Nevertheless, users are advised to unlink their cameras from Google Nest Hub until patches arrive.

The takeaway for our podcast is to always keep in mind that in 2019 and 2020, cloud connection remains a high-risk business. It just is. Inherently. Perhaps we will eventually learn how to secure these systems. But it's very clear that we're not there yet, and that products and services are being launched and offered with inadequate attention paid to security. Whether you're outsourcing management functions to a third party who then becomes a conduit to carry destructive ransomware into and throughout your organization, or using cloud-connected surveillance cameras which lord-only-knows-who can also monitor, there's no arguing that there absolutely are significant upside benefits. But so far they don't come without some significantly balancing downside risks.

And while we're on the topic of cloud-based risks...
Microsoft's security blog posting of December 18th carried the title: “Data science for cybersecurity: A probabilistic time series model for detecting RDP inbound brute force attacks”

https://www.microsoft.com/security/blog/2019/12/18/data-science-for-cybersecurity-a-probabilistic-time-series-model-for-detecting-rdp-inbound-brute-force-attacks/

I'll share the introduction so that we can see what the intent was:

Computers with Windows Remote Desktop Protocol (RDP) exposed to the internet are an attractive target for adversaries because they present a simple and effective way to gain access to a network. Brute forcing RDP, a secure network communications protocol that provides remote access over port 3389, does not require a high level of expertise or the use of exploits; attackers can utilize many off-the-shelf tools to scan the internet for potential victims and leverage similar such tools for conducting the brute force attack.

Attackers target RDP servers that use weak passwords and are without multi-factor authentication, virtual private networks (VPNs), and other security protections. Through RDP brute force, threat actor groups can gain access to target machines and conduct many follow-on activities like ransomware and coin mining operations.

In a brute force attack, adversaries attempt to sign in to an account by effectively using one or more trial-and-error methods. Many failed sign-ins occurring over very short time frequencies, typically minutes or even seconds, are usually associated with these attacks. A brute force attack might also involve adversaries attempting to access one or more accounts using valid usernames that were obtained from credential theft or using common usernames like “administrator”. The same holds for password combinations. In detecting RDP brute force attacks, we focus on the source IP address and username, as password data is not available.

In the Windows operating system, whenever an attempted sign-in fails for a local machine, Event Tracing for Windows (ETW) registers Event ID 4625 with the associated username. Meanwhile, source IP addresses connected to RDP can be accessed; this information is very useful in assessing if a machine is under brute force attack. Using this information in combination with Event ID 4624 for non-server Windows machines can shed light on which sign-in sessions were successfully created and can further help in detecting if a local machine has been compromised.

In this blog we’ll present a study and a detection logic that uses these signals. This data science-driven approach to detecting RDP brute force attacks has proven valuable in detecting human adversary activity through Microsoft Threat Experts, the managed threat hunting service in Microsoft Defender Advanced Threat Protection. This work is an example of how the close collaboration between data scientists and threat hunters results in protection for customers against real-world threats.

Okay, so here's what we know now that we didn't know before...

Microsoft learned that approximately 0.08% of RDP brute-force attacks are successful, and RDP brute-force attacks last 2-3 days on average. For the study, Microsoft collected data on RDP login-related events from more than 45,000 workstations running Microsoft Defender Advanced Threat Protection which is the commercial version of its built-in free Defender antivirus app.

The data was gathered across several months, and involved collecting details about both failed and successful RDP login events, reflected by Windows log events with IDs of 4265 and 4264, respectively, and including the usernames a user or attacker used.

As we know, RDP is often deployed in enterprise environments to allow system administrators to manage servers and workstations in remote locations, and it is also used by employees to access their machines when away traveling.

We also know (horrible authentication bypass mistakes notwithstanding) that atackers mount attacks against Windows systems with open RDP ports. These attackers employ standard credential-stuffing attacks to brute-force cycle through many username/password combinations, attempting to guess the target computer's RDP login credentials. These attacks typically use combinations of usernames and passwords that have been leaked online after breaches of online services. Massive lists of usernames and passwords are easily obtained and used.

Microsoft says that recently observed RDP brute-force attacks typically last 2-3 days on average, with around 90% of cases lasting one week or less, and fewer than 5% lasting two weeks or more. The attacks tended to last for days rather than hours because attackers were deliberately throttling their access in an attempt to avoid having their IPs detected as malicious and therefore banned by defensive firewalls. Consequently, rather than blasting an open RDP port with thousands of failed attempts at once, attackers try only a few combinations to almost literally, slip under the radar.

As I noted earlier, Microsoft said that "Out of the thousands of machines with RDP brute force attacks detected in our analysis, we found that about .08% were compromised." I did the math, and at a success rate of .08%, that's 36 machines out of 45,000.

The Microsoft research team added that "... across all enterprises analyzed over several months, on average about 1 machine was detected with high probability of being compromised resulting from an RDP brute force attack every 3-4 days." They noted that "A key takeaway from our analysis is that successful brute force attempts are not uncommon; therefore, it's critical to monitor at least the suspicious connections and unusual failed sign-ins that result in authenticated sign-in events."

As our listeners know, I manage a number Windows servers remotely, but you won't find a single instance of RDP instance listening on **any** publicly exposed port. It should be as clear as day that there's just no safe way to have RDP publicly exposed -- ever. Tuck it safely behind a VPN with a multi-factor login including TOTP and a client credential certificate. Then the entire problem just goes away.

But, perversely, Microsoft concludes their otherwise useful and interesting research report by recommending that system administrators combine and monitor multiple signals for detecting RDP inbound brute force traffic on a machine. According to Microsoft, such signals should include:

- Hour of day and day of week of failed sign-in and RDP connections
- Timing of successful sign-in following failed attempts
- Event ID 4625 login type (filtered to network and remote interactive)
- Event ID 4625 failure reason (filtered to %%2308, %%2312, %%2313)
- Cumulative count of distinct username that failed to sign in without success
- Count (and cumulative count) of failed sign-ins
- Count (and cumulative count) of RDP inbound external IP
- Count of other machines having RDP inbound connections from one or more of the same IP

For what it's worth, I think that approach is totally nuts. It would be like saying to the vendor of a truly impenetrable security door: "Oh, I know that you doors are impenetrable and inexpensive, and that the door we have here is kinda fragile, flimsy and is falling apart. But we're just going to ask our receptionist to keep an eye on it to be sure that no one she doesn't know comes through." [then] "Oh... You didn't see our receptionist when you came in? No, she doesn't always feel well on Mondays."

POS - Point Of Sale - systems continue to be a huge problem.
These hacks are less technically interesting. And credit card theft typically represents widely distributed, though individually recoverable, pain. As I well know, having had more than my share of credit cards cancelled and reissued after incidents of online fraud, consumers are protected from any fraudulent charges on credit cards. So the only cost to me has been the annoyance. But I don't believe that's the case with debit cards where the customer's money may be gone for good. We've said here before that if given a choice between using a debit or a credit card, a credit card provides significantly greater protection from online fraud.

What's interesting is that purchasing food with a card at a restaurant would not traditionally have been thought of as "online." But, of course, everything is computerized and online today.

In any event, during our "Decade of Hacks" retrospective two weeks ago Leo and I remembered the history making attack on the Target chain of retail stores. I wanted to remind everyone these attacks have not stopped or even slowed down. But another major restaurant chain owner, Landry's, recently revealed an attack on the POS systems of its restaurants which affected the many different of its chains.

When I heard "Landry's" I didn't think much of it, since I've never eaten at a "Landry's Restaurant" -- or so I thought. Then I looked over the chains that they own and operate and I recognized many. It turns out that Landry's owns and operates more than 600 bars, restaurants, hotels, casinos, food and beverage outlets with over 63 different brands such as Landry's Seafood, Chart House, Saltgrass Steak House, Claim Jumper, Morton's Steakhouse, Mastro's Restaurants, and Rainforest Cafe.

According to the breach notification they published this week, the malware was designed to search for and steal sensitive customer credit card data, including credit card numbers, expiration dates, verification codes and, in some cases, cardholder names. The PoS malware infected point-of-sale terminals at all Landry's owned locations. They said that the end-to-end encryption technology used by the company prevented attackers from stealing payment card data from cards swiped at its restaurants. But Landry's outlets also use "order-entry systems with a card reader attached for waitstaff to enter kitchen and bar orders and to swipe Landry's Select Club reward cards," which allowed attackers to successfully steal customers' payment data "in rare circumstances" when waitstaff mistakenly swiped payment cards on them... Whatever all that means. Perhaps the point was that it wasn't as bad as it might have been.

According to the company, the POS malware was actively scanning their systems between 13th March 2019 and 17th October 2019 for swipe cards; and at some locations, it may have been installed as early as 18th January 2019. They said that "During the investigation, we removed the malware and implemented enhanced security measures, and we are providing additional training to waitstaff."

And, also...

The US convenience store chain "Wawa" said last month that it recently discovered malware that skimmed customers' payment card data at just about all of its 850 stores.

The infection began rolling out to the store's payment-processing system on March 4 and wasn't discovered until December 10, an advisory published on the company's website said. It took two more days for the malware to be fully contained. Most locations' point-of-sale systems were infected by April 22, 2019, although the advisory said some locations may not have been affected at all.

The malware collected payment card numbers, expiration dates, and cardholder names from payment cards used at "potentially all Wawa in-store payment terminals and fuel dispensers." The advisory didn't say how many customers or cards were affected. The malware didn't access debit card PINs, credit card CVV2 numbers, or driver license data used to verify age-restricted purchases. Information processed by in-store ATMs was also not affected. The company has hired an outside forensics firm to investigate the infection.

And this disclosure last month came after Visa issued two security alerts—one in November and a second in December—warning of payment-card-skimming malware at North American gasoline pumps. Card readers at self-service fuel pumps are particularly vulnerable to skimming because they are a bit behind the times, reading payment data from cards' magnetic stripes rather than card chips, which are less susceptible to skimmers. Visa's first advisory in November said:

The recent attacks are attributed to two sophisticated criminal groups with a history of large-scale, successful compromises against merchants in various industries. The groups gain access to the targeted merchant's network, move laterally within the network using malware toolsets, and ultimately target the merchant's POS environment to scrape payment card data. The groups also have close ties with the cybercrime underground and are able to easily monetize the accounts obtained in these attacks by selling the accounts to the top tier cybercrime underground carding shops.

So, although these POS attacks might not be super-sexy, fun, or particularly imaginative, they are quite sophisticated and lucrative for those who manage to worm their way in and pull them off.

Be careful when you flip that switch!
RuNet disconnection tests were successful, according to the Russian government.

Monday of Christmas week, the Russian government announced that it concluded a series of tests during which it successfully disconnected the country from the worldwide internet. The tests were carried out over multiple days, starting the previous week, and involved Russian government agencies, local internet service providers, and local Russian internet companies. The goal of the tests was to determine and verify whether the country's national internet infrastructure -- known inside Russia as RuNet -- could function without access to the global DNS system and the external internet.

After they pulled the switch, all Internet traffic was rerouted internally, effectively making Russia's RuNet the world's largest intranet. The government was not interested in revealing any technical details about the tests and what exactly they consisted of. It only said that the government tested several disconnection scenarios, including a scenario that simulated a hostile cyber-attack from a foreign country.

As was cited by multiple Russian news agencies, Alexei Sokolov, deputy head of the Ministry of Digital Development, Communications and Mass Media said: “It turned out that, in general, that both authorities and telecom operators are ready to effectively respond to possible risks and threats and ensure the functioning of the Internet and the unified telecommunication network in Russia.” Alexei said the results of the tests would be presented to President Putin in 2020.

We've discussed this plan several times in the past, so we already understand its outline. The tests were the result of many years of planning, including some required law-making by the Russian government as well as physical modifications to Russia's internal internet infrastructure. The tests had initially been scheduled for April of 2019, but were delayed until this fall, to give the Kremlin more time to pass their new “Internet Sovereignty” law which grants the Russian government the power to disconnect the country from the rest of the internet at will and with little explanation, on the grounds of "national security."

To enable this, the law mandates that all local internet service providers re-route all internet traffic through strategic choke-points under the management of Russia's Communications Ministry.

These chokepoints serve as a gigantic kill-switch for Russia's connection to the external Internet. But, perhaps significantly, they also provide a powerful opportunity for Internet surveillance, not unlike China's Great Firewall technology.

It's truly difficult in this day and age to consider having the US being disconnected from the entire rest of the world... But I can see its possible value as a last resort.

Python 2.7 Reaches End of Life After 20 Years
How quickly 20 years passes. Python v2.7 was released in the year 2000. Python 3 was released six years later in '06. But, as we so often see with technology, "if it's not broke, leave it alone." Python 2.7 was working quite nicely, thank you very much, so why should we move to 3? And, sure enough, due to 2.7's incredible popularity, and even though 2.7 had its original sunset retirement planned for 2015, the Python developers decided to continue supporting both development branches.

But I'm sure it hasn't escaped anyone's attention that it's now the year 2020. And, yes, after a good 20 year run, Python 2.7's retirement day finally arrived 6 days ago. Python 2.7 has officially reached its end of life. And, as we know, painful as this will be, this will ultimately be "a good thing."

Dropping 2.7 will allow the Python team to focus their full attention on Python 3 with an eye toward increasing its speed and patching any bugs.

In their announcement they wrote: "We are volunteers who make and take care of the Python programming language. We have decided that January 1, 2020, will be the day that we sunset Python 2. That means that we will not improve it anymore after that day, even if someone finds a security problem in it. You should upgrade to Python 3 as soon as you can."

That said, the team does plan one last release of Python 2.7 in April. That final release will include any final bug and security fixes from 2019 and any more up until that final release date. They want to lave it as stable and solid as possible.

But it's very clearly time to move on. Bleeping Computer noted that for those who really do require Python 2.7 compatibility, and for whom moving to 3 is a problem for whatever reason, an option would be to move to PyPy. PyPy is an alternative re-implementation of Python written around a JIT (a just in time) compiler rather than an interpreter. As a result, Python programs often execute faster under PyPy than interpreted Python. PyPy compiled programs also tend to require less memory to execute.

Most Linux distros are working to migrate their dependencies and libraries from 2.7 to 3.

Debian, Ubuntu and Fedora have begun the process of updating Python 2 libraries to their Python 3 equivalents. The Debian "Buster" (10x) release, and Ubuntu 18.04 LTS will both be using Python 3 as their default, but 2.7 will still be available for those wishing to install it. In the current release of Fedora 31, Python 3.6 is already the default version installed.

Red Hat has stated that even though the Python Software Foundation (PSF) has retired Python 2.7, they will continue to support it through the normal RHEL lifecycle. So, for another 4 years.

HackerOne's 20 top bug bounty programs
As our listeners know, we are quite bullish about the emergence of a new category of employment: Security vulneratility hunting and reporting for profit. This part- or full-time career path is enabled by companies such as HackerOne, my personal favorite since, unlike Zerodium, HackerOne does not hoard then turn around and secretly selling their submissions into the grey market for potentially malign use by national law enforcement, intelligence and cybercrime entities. HackerOne simply manages the bounty programs offered by legitimate firms who have grown up and understand that putting new eyes on their code only makes sense and that incentivizing such scrutiny requires bounty payouts.

So it's instructive to check-in every so often on the Top 20 HackerOne clients to see how their bounty programs are going.

https://www.hackerone.com/sites/default/files/2019-06/h1-718-top-20-public-bug-bounty-programs.pdf

Ranked ranked downward from the most money paid, the top 20 companies are:

Verizon Media / Uber / PayPal / Shopify / Twitter / Intel / Airbnb / Ubiquiti Networks / Valve / GitLab / GitHub / Slack Starbucks / Mail.ru / Grab / Coinbase / Snapchat / HackerOne / DropBox / VK

Verizon is solidly at the top having paid out more than $4 million since launching their program back in February of 2014. They are #1 in all-time bounties paid, in the most hackers thanked, and in the most reports resolved.

The #2 slot is Uber with less than half the total bounties paid at $1.8 million, but whereas Verizon's top bounty ever paid was $6K, Uber has paid $15k. And, also, Verizon's $4M are spread across 5269 reports, whereas Uber's $1.8M is spread over only 1172 reports. So it does appear that Uber pays bigger bounties.

But PayPal, in the #3 slot takes the award for the single largest bounty paid through HackerOne, at $30K. And they are also clearly paying more per bounty since their all time total payout is $1.25M, but that's spread across only 430 reports. PayPal has only been a participant on HackerOne since 2018, but they are in the top 5 fastest response time, so they clearly see this as a security win for them.

I won't keep enumerating every details of the next 17, all of the details are available through the link in the show notes. But I'll note that Shopify boasted the fastest time to bounty payout of only 2 days and is among the top 5 in most reports resolved and largest single bounty paid. Valve is also among the top five paying the single largest bounty, as are GitHub, Coinbase, Snapchat, DropBox.

So, just another reminder that it's possible to pay the bills while truly doing good, actively and objectively improving the overall security of this industry.

This week's possible action item:
As we often do, we do have one potentially significant and widespread new vulnerability to inform our listeners of: The prolific Tencent Blade Team recently disclosed another batch of serious SQLite vulnerabilities they have named "Magellan 2.0."

Newly discovered vulnerabilities in the extremely popular and widespread SQLite database engine affect a wide range of applications that utilize it as a component within their software packages. The SQLite RDBMS is used pretty much everywhere, including by a wide variety of programs including Google Chrome, Mozilla Firefox, Windows 10, and, as I said, just about everything else.

It’s been nearly a year since the Tencent Blade Team disclosed their original Magellan 1.0 SQLite vulnerabilities. These new revelations affect all programs that utilize SQLite as a component in their software which allows external SQL queries.

One of the biggest target is Google's Chrome browser. Tencent's advisory wrote: "These vulnerabilities were found by Tencent Blade Team and verified to be able to exploit remote code execution in the Chromium render process." The advisory continues: "As a well-known database, SQLite is widely used in all modern mainstream operating systems and softwares, so this vulnerability has a wide range of influence. SQLite and Google had confirmed and fixed these vulnerabilities. We will not disclose any details of the vulnerability at this time, and we are pushing other vendors to fix this vulnerability as soon as possible."

Using these vulnerabilities, Tencent was able to remotely execute commands in Google Chrome as long as WebSQL was enabled in the browser. This is a critical vulnerability as it means remote attackers could potentially use this vulnerability to fully compromise a computer.

The advisory states: "If you are using a software that is using SQLite as component (without the latest patch, which is 13 Dec 2019), and it supports external SQL queries. Or, you are using Chrome that is prior to 79.0.3945.79 with WebSQL enabled, you may be affected. Other devices such as PC/Mobile devices/IoT devices may also be affected, depends on if there's a proper attack surface."

Tencent has not seen any indication that these vulnerabilities have been utilized in the wild and reported them to Google and SQLite back on November 16th, 2019.

After reporting the vulnerabilities, they were assigned five CVE's (CVE-2019-13734, CVE-2019-13750, CVE-2019-13751, CVE-2019-13752, CVE-2019-13753) and were fixed in Google Chrome 79.0.3945.79 and in patches applied to SQLite on December 13th, 2019.

However... And this is the potentially crucial part for our listeners: SQLite is everywhere. The danger is reduced if software using SQLite is not making its services available publicly. Presumably, anyone behind a NAT router is safe, since nothing is public unless UPnP has been asked to open a port for unsolicited incoming traffic. But, ideally, the news of these problem will filter down to =ALL= the SQLite-using software and it will be updated to eliminate these problems. In the meantime, keep this in the back of your mind.

SCI-FI

- **Danger Will Robinson!! Danger!!**
    - On Christmas Eve Netflix dropped the second 10-episode season of "Lost in Space."
- **The Mandalorian**

SQRL for Drupal

Jürgen Haas: "Happy to announce that I was finally able to finish off the SQRL integration into Drupal 8 (and the forthcoming version 9). It is feature complete and supports all of the SQRL protocol version 1.

A test server can be found at https://sqrldemo.lakedrops.com and the source code with the Drupal module is available from https://www.drupal.org/project/sqrl - the installation on any existing Drupal 8 site is as simple as "composer require drupal/sqrl" followed by "drush en sqrl" and that's all to get it up and running.

Maybe a new forum section on sqrl.grc.com might be worth having?

To me this is a great start into 2020 and I wish everybody around here a Happy New Year too.

((I thanked Jürgen for all his terrific work, told him that I'd be mentioning this on this week's Security Now, and that I would immediately setup a forum at SQRL.grc.com for him to moderate for the public management of this great work.))

Shortly afterward, “Brian of London” chimed in by posting:

> And as one of the first testers, I can confirm it was ridiculously easy to fire up SQRL client (can't even remember which client I used) sign in for the first time and then set up an Avatar and so on. I haven't given it an email or password, not needed.
>
> Brian of London

SpinRite

As I mentioned at the time, when I formally turned the SQRL project over to its terrifically capable community, this past holiday period was my time to give some long-needed attention to GRC's servers at Level3. Among the work I did, our long-running FreeBSD v5 UNIX DNS and newserver hardware was replaced after 15 years of flawless service. Nothing was wrong with it even after all that time, but I wanted to move us from 32 bits to 64, update to a newer version of BIND for serving DNS, run newer OpenSSL, and so forth. I am still wrapping up the re-addition of some of the deep and useful customizations I made to the old newsserver code. Then I'll update our Windows servers with OS updates and updates to a range of packages running on the isolated forums server.

I want to get all of that completely behind me before I switch back to full-time work on SpinRite... which I am SO excited to be getting back to!

Our Malware Lexicon

We're always talking about this or that sort of threat. And just like biological viruses, malware evolves over time. Some, are highly opportunistic, rising briefly to capture our attention while capitalizing upon a short-term opportunity. But many others have evolved to exploit more fundamental flaws and problems that we haven't yet, as an industry, resolved. The ongoing troubles with Point of Sale systems we were talking about earlier are a good example of an early problem that persists, and the new and clearly justified worries over the security of IoT are a good example of a relatively recent new threat.

So, when I ran across a nice summary breakdown of malware methodology by Sophos, I thought that an appropriate way to ring in the New Year would be to quickly run through the lexicon of terminology we now use to describe the various sorts of enduring threats which continue to evolved and mature.

So, what are the deadly seven?

#1. KEYLOGGERS

Keyloggers are surprisingly simple, and can be implemented in many different ways. They hook into the stream of data coming from our keyboards which allows them to capture everything we type.

As anyone who has seen one of my SQRL presentations knows, I often use the example of needing to log into my SouthWest Airlines account at a Las Vegas hotel's business center and being horrified by the idea of entering my username and password into the widely shared PC there. (SQRL, of course, allows for a zero keystroke login to untrusted PCs.)

And keyloggers often don't merely know "you typed F" – they get enough detail to tell that you pressed the left Shift key down, then depressed F, then released F, then let go of the shift. That means they can even keep track of keystrokes that don't produce any visible output, such as function keys, backspaces and other key combinations that turn options on or off. And keyloggers don't need to be implemented down at the operating system level, and they often don't need administrative or root powers to hook themselves into the keystroke data stream. Anyone who has installed a keystroke macro recording and playback tool -- or an on-the-fly spell check and correct -- will realize that the application sitting in their system tray is able to monitor everything they are typing.

Moreover, JavaScript running in our browser can monitor (and could alter, if it had some reason to -- such as delaying your own login) the flow of keystrokes as you browse. That means that rogue JavaScript injected into a login page could recognise and steal usernames and passwords. And let's recall that the integrity of our HTTPS connections is being subverted by anything we allow to "intercept" and "inspect" our data -- whether corporate or locally installed A/V.

Banking Trojans commonly include a keylogger module so they can try to capture our passwords when they recognise that we're logging in to our bank. And, as we know, keyloggers can be hardware – a small device that’s just a lump in the cord connecting the physical keyboard to the computer. In fact, keyloggers have been completely enclosed in the USB plug at the end of the keyboard's cord. And such hardware loggers cannot be readily detected by software and, of course, they survive a full system cleaning and reload.

#2. DATA STEALERS

Data stealers is the somewhat generic name for any malware that gets into our machine and then goes hunting around our hard disk, and perhaps even around our whole network if it can, looking for files that contain data that’s worth something to crooks.

Once upon a time -- ah, those quaint vintage days of computing before the Internet -- we mostly had true computer viruses which spread automatically by theselves, often by spewing out emails containing an infected attachment. Back then, many viruses would search through qualifying files on our computers, looking for text strings that matched a specific pattern such as as eMail address. By harvesting email addresses from everywhere, not just from our email software, they could come up with exensive lists of potential new victims, even people we'd never directly contacted, but whose addresses appeared in documents, marketing material, or saved website pages.

But times have changed. Contemporary data stealers are much more sophisticated and wide-ranging. They search out and find bank account details, ID numbers, passport data, credit cards and account passwords. They know how to recognise special files by their names or their internal structure, including password vaults that contain our credentials and browser databases that may contain tell-tale data such as authentication tokens and browsing history. Several of the other classes of malware such as Bots and Banking Trojans include sophisticated data stealing modules as another means for obtaining value from their victims.

#3. RAM SCRAPERS

Malware can’t always find what it wants in files on our computer even if the malware has admin or root access. That’s because useful data might only ever exist temporarily in memory before being deliberately scrubbed without ever being written to disk.

Permanent storage of some data is now prohibited by regulations such as PCI-DSS, which is the Payment Card Industry Data Security Standard, and by the GDPR, Europe's General Data Protection Regulation. Those regulations prohibit the permanent storage of some information such as the CVV number used to authorize a credit card payment. Such regulations is naturally bad news for cybercrooks, since it means they can’t easily get hold of those codes for transactions that have already occurred.

…but with RAM scraping malware that keeps an eye on data as it is stored temporarily in memory, attackers may be able to spot such critically useful data, such as CVVs and the accompanying full credit card information and “scrape” it right out of RAM.

Since computers MUST have, for example, a private key in RAM in order to perform decryption, valuable secret data MUST transiently exist in RAM -- even if only very briefly. So things such as decryption keys, plaintext paswords and website authentication tokens are being watched for by RAM scrapers.

#4. BOTS

A "bot" is the term we have given to malware which sets up shop in a machine then phones home to its "bot master" to report in, request and await further instructions. It essentially establishes a semi-permanent backdoor into a computer so that the bad guys can send commands from wherever they are.

And we call a collection of such bots, a botnet. The other popular term for "Bot" is "Zombie" because they can also act a bit like sleeper agents. The commands bots understand including sending boatloads of spam from your IP address, searching for local files, sniffing passwords, blasting other machines on the Internet with floods of traffic, and even clicking online ads to generate pay-per-click revenue. Modern bots are also capable of receiving updates and downloading new and improved modules. We've recently talked about the evolving strategies bots are using to sneak these updates into a network right under the noses of watchful A/V screeners.

And since bots are initiating connections outward to their bot masters, just like any IoT devices or other cloud-based appliances, their communications are not automatically blocked by our trusty home routers which dutifully ignore all unsolicited inbound traffic.

#5. BANKING TROJANS

Banking Trojans deserve their own sub-class of malware due to their specialization. They exclusively target their victim's online banking information. And, as we night imagine, banking trojans typically include a keylogger component, to sniff out passwords as they are entered. They also often incorporate a data stealer component to trawl through likely files such as browser databases and password vaults in the hopes of finding unencrypted passwords or account details.

A trick widely used by banking trojans is web form injection, where the malware adds additional data fields to forms displayed in the browser. By doing this they hope to trick their victims into entering additional unnecessary data, such as a credit card number for "verification" or a date of birth. The typical computer user has completely given up any hope of actually understanding what's going on. They just hope for the best. So when their bank suddenly asks for a credit card number for some reason, they just think... uh, okay I guess.

#6. RATS

The RAT -- short for Remote Access Trojan -- has much in common with a "bot", but it differs in that it's not usually part of a massive campaign to see how many "bots" can be corralled and managed for mass attack events. Instead, a RAT enables more targeted and potentially damaging intrusion. They can take screenshots, listen to our rooms audio through the PC's microphone and turn on our webcams.

Finally, last but certainly not least, we have...

#7. RANSOMWARE

After 2019, it would likely be no exaggeration to state that ransomware is probably the most feared type of malware. Whereas many people might not know exactly that a Bot or a RAT are, pretty much everyone has heard horrifying stories of entire municipalities, companies, or health care providers being shut down by ransomware. They may not know exactly what it is, but they know it's a current problem for some reason.

As we've observed here last year, the enabling factor for ransomware was the rise of readily exchangeable cybercurrency. Until there was a safe and untraceable means for receiving payment, the ransomware "business" wasn't really able to take off. But, today, there are even companies specializing in being the go-between between a ransomware victim and the extortionists who are demanding payment. This has solved the "I don't know how to send anyone bitcoins" problem.

Ransomware attacks can be SO lucrative that they are the inverse of the scattershot "Bot" model. Attackers can afford to take their time to maximize the damage done. After finding their way into a victim's network they setup to scramble hundreds or even thousands of computers at the same time. And even if offline backups exist, the time and labor required to reimage and restoring thousands of computers has been shown to be hugely damaging all by itself. And if backups are NOT securely offline they are likely to fall, too, making restoration impossible.

And because the payoff might be significant, the attackers can afford to spend the time to research the entire cybersecurity countermeasure situation to sidestep or disable anything that might halt or limit the ransomware's reach and effect.

So, this first podcast of 2020 launches us into what is certain to be a very interesting year of cybersecurity events... beginning with next week's final Windows 7 Patch Tuesday update.

Stay Tuned!