Our Malware Lexicon

Description: 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, examine 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.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-748.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-748-lq.mp3

SHOW TEASE: This week on Security Now! - should I say "this decade"? We'll start the show with a little decade talk. Steve breaks down the lexicon, the terminology associated with malware. Super interesting stuff there. Also standards around warrant canaries; how Xiaomi cameras actually sent random private video feeds to Nest Hub users, not a good look; HackerOne's top 20 bug bounty programs; problems with point-of-sale systems; and a whole lot more, coming up next with Steve Gibson on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 748, recorded Tuesday, January 7th, 2020: Our Malware Lexicon.

It's time for Security Now!, the '20s edition. I'm Jason Howell, filling in for Leo Laporte, who's in Vegas. Joining me as always, Steve Gibson, the amazing Steve Gibson. How are you doing, Steve?

Steve Gibson: The indefatigable Steve Gibson.

JASON: Why aren't you in Vegas for CES?

Steve: You know, this is one of the first things that - okay. So first of all, I don't think I ever missed a COMDEX back in the day.

JASON: Ah, I remember COMDEX, yeah.

Steve: That was the big computer dealer expo. That was the PC industry show. And I did a CES when I was with Koala, after selling them the light pen. They did the KoalaPad back in the early days, an early touch stylus. And I've been to a bunch of CESes. That used to be a thing. But for me, what the Internet does is free you from the need to go to all these things.

JASON: True.

Steve: And I hear people saying, "Oh, Gibson's not a real security guy. He never goes to conferences." It's like, no. I mean, yeah, I get it that it's nice to meet people and shake hands and hobnob and, you know. And if I were still in my 20s and 30s I'd probably think that was really cool. But we have a complete seat to CES just by looking at what everybody is posting about CES. You'll end up with a much better sense for the show than if you tried to do it on foot. And besides, I'm busy. People would much rather have me finishing up the work I'm doing on revamping our servers at Level 3, preparing to get into the return to work on SpinRite, than wandering around and ending up with sore feet at the end of a long day of CES. So priorities.

JASON: Completely understand. Completely agree. I've had the years of going to CES year after year after year and running on that treadmill endlessly. And then I've had of course in the last however many years here at TWiT, we've really taken many of these years kind of the "sit back and view it" approach. And you're right. It's kind of like the perfect example of seeing the forest for the trees; right? Like when you're at CES, you're seeing all the trees, and it's really cool. It's good for one thing because you can have an up-close experience with some of these things that we hear about. But standing back, there's benefit there, too, because you just get this perspective of everything on equal footing. Some of the noise is gone, so it's kind of like you see it for what it is. I don't know, there's benefits to both sides. I suppose it really depends.

Steve: Well, I also don't subscribe any longer to lots of paper magazines. Popular Electronics, Popular Science, Scientific American, I mean, I used to do that, too. But then the Internet happened, and like, hello, this is better. So, yeah. I'm glad they're doing the shows. I'm glad Leo and a camera crew are there, and TWiT is going to bring us CES.

JASON: That's right, that's right. They're churning out a whole bunch of content to find. And of course TWiT.tv you'll find all the content published on the site here.

Steve: So, yeah, of course. So anyway, I thought for the first episode of this, whatever it - okay. You and I were talking before the show that technically this is not the beginning of a new decade. This is like the "are you counting from zero or one" problem that bites so many programmers in the butt throughout their entire programming history.

JOHN SLANINA: I'm sorry, Steve. I'm sorry, Steve, I have to interrupt. I really feel strongly about this.

Steve: Okay.

JOHN: A decade is 10 years. You can start a decade at any point in time.

JASON: That is true.

JOHN: What you cannot say is this is the 201st decade. That is the only thing you cannot say.

JASON: Ah, okay.

JOHN: It's not the beginning of the 201st decade until next year. But any other - you can start a decade any time you want.

JASON: We can start a decade right now.

Steve: We are, for example, in the second decade of this podcast.

JASON: Woohoo! Yes. That makes sense. That is true.

Steve: So anyway, I think it was Sophos who did sort of an interesting - they called it a virus, and they're not really viruses. It's more generically malware. But they divided all of the malware that we know into seven categories. And so I thought it would be interesting at the end, after catching up with a bunch of interesting news of the week, as we always do, to just sort of scan through this division. I called it "the lexicon," so this podcast is titled "Our Malware Lexicon." That is, the common sort of aggregation of malware that has now sort of settled upon us. So that's how we're going to end things.

But we're going to first take a look at a really interesting proposed standard for creating machine-readable warrant canaries. And we'll talk a little bit about what warrant canaries are and how they happened, also, just to give our readers a little bit of background. We also have a precautionary lesson about a big Xiaomi blunder. We're going to examine Microsoft's research into brute-forcing remote desktop protocol and why they still don't have it right.

We're going to look at the continuing problem at the point of sale, follow up on Russia's plan that we've been watching and following for the last few years to disconnect itself or be able, at least, to disconnect itself from the Internet. We're going to look at the end of life of the beloved Python v2.7; review the top 20 HackerOne bounty payers; warn our listeners, as we often do, there's like one definite oops, pay attention to this one, bad new SQLite security vulnerabilities; cover just quickly a little bit of sci-fi, SQRL, and SpinRite miscellany; and then take a look at our malware lexicon. So buckle up. I think we're going to have a good podcast.

JASON: Buckle up. If today is any indication...

Steve: As we kick off whatever this is, the decade or the who knows what.

JASON: I think we now know it is the decade. It is the decade that we have defined because anything could be a decade.

Steve: So speaks John from his booth of authority, yes.

JASON: And I love that he did that because as far as I'm concerned, JammerB had the right answer for that question. That's how I feel. And I also appreciate that we can now say it's "the 20s" because for all of my life "the 20s" has felt old-timey.

Steve: The Roaring Twenties.

JASON: And now it's like the future. So that's kind of weird. All right. Do we start with this picture? We've got a little picture here. It's a pretty picture.

Steve: Yes, we do. I just sort of grabbed it. I thought it was kind of a cute logo which was part of this - it's called the CanaryTail on GitHub. It's their logo for the page to deal with this proposed machine-readable warrant canary. So I just thought that was a great Picture of the Week since it ties in with our first story. Of course the term "canary," as we know, comes from the idea of taking a canary down into a coal mine as a low-tech oxygen and noxious gas monitor.

Wiktionary defines "canary in a coal mine" as "something whose sensitivity to adverse conditions makes it a useful indicator of such conditions, something which warns of the coming of danger or trouble by a deterioration in its health or welfare." And I guess actually for some people bunions are supposed to be able to tell you of an oncoming

storm or something; right? Anyway, I don't know if that quite qualifies as a warrant canary. But okay.

So the trouble was that government warrants compelling the release of information were typically accompanied by gag orders enjoining the party served with the warrant from making any disclosure about being in receipt of such warrant. So it was you must provide us with the following information, and you cannot tell anyone that we asked you and you provided. This was passed in 2001 as part of the U.S. Patriot Act, and it enabled authorities for the first time to ask for personal information stored by service providers about U.S. citizens and also allowed them to issue gag orders that would prevent the organizations from telling anyone about it. So it meant that government could essentially access an individual's private information without that person knowing.

And, you know, you can understand why the government would want that. That was, for me, that was like sort of the first writing on the wall that encryption was in trouble because we can pass whatever laws we want to in this country, and then that's the law, as they say. And, boy, I mean, Congress is sure dragging their heels. Law enforcement - well, anyway, I won't get onto the encryption debate. We certainly spent a lot of time covering that in 2019.

In this case, privacy-respecting companies like, well, anybody, ISPs, any kind of service provider, anybody who holds people's data and wants them to be able to trust it, want their users to know whether the government is asking for the information that they've promised not to provide. So the idea of a warrant canary, I did a little bit of background research because I was curious, was first conceived by a guy named Steve Schear in 2002 after the Patriot Act came into effect. And his idea, which I think is sheer brilliance, was to provide a passive way, rather than an active way, which was illegal, for warning people that an organization holding their data has received a subpoena.

So instead of telling people that they had been served with a subpoena, which became illegal under the Patriot Act, his clever idea was to reverse that and stop telling them, instead, that they had not been served with a subpoena, which apparently was not illegal because I don't think you could compel someone to say something. You could just tell them what they can't say. So it was very cute.

Anyway, so to do this, the organization would typically display a public statement online that it would only change if the authorities were to serve it with a warrant. And so as long as the statement stays unchanged, individuals checking it would know that their information was safe. So if a "nothing to see here, move along" statement were to change or disappear, then people could infer that not all is well, so the organization doesn't have to explicitly say so.

So that idea is clever, but until now it has not scaled well. Hopefully that may be about to change. But there can be problems. A classic example is what happened about a year and a half ago with SpiderOak, that is, a high-reputation neat provider of cloud-based encryption services, during the summer of 2018. They essentially were forced to walk people back from the ledge after concerns were raised because they changed their warrant canary.

On August 18th of 2018 they posted on Medium hopefully a clarification. They said: "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," they wrote, "that some people are concerned that this is a signal that we have in some way been compromised." So they said: "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 this case it was four years ago - "in August of 2014."

Then they said separately, on a separate line, "We have received zero search warrants, zero subpoenas, zero court orders, and zero national security letters. Even better for our customers, we could not hand over their data even if we were asked to. The No Knowledge approach that SpiderOak uses" - that's the so-called TNO, Trust No One approach of pre-encryption technology - "means that we don't have the keys to decrypt the data you trust us to store for you." And they finished: "Thank you for choosing SpiderOak and deciding to trust in cryptography instead of promises." And of course we know that's my philosophy, as well.

But anyway, that highlights the problem of there not being a standard. And what happens even with a well-meaning change can be like, oh my god, oh my god; you know? Does that mean the canary died? It's like, okay, no. Okay. So there is now a proposed standard on GitHub. It was put up by a GitHub user using the moniker "carrotcypher," C-A-R-R-O-T-C-Y-P-H-E-R. And it was inspired, that is, this proposed standard, inspired by the work of organizations like the Calyx Institute, which is a technology nonprofit that develops free privacy software and what is now a defunct Canary Watch project from the EFF, the Electronic Frontier Foundation. Also the Freedom of Press Foundation, NYU Law, Calyx, and the Berkman Center.

Anyway, so 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" - which the EFF believes they are - "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." Just like we saw in the case of SpiderOak.

So the hope of this CanaryTail is to change this. As it explains on its GitHub readme.md page, they said: "We seek to resolve those issues through a proper standardized model of generation, administration, and distribution, with variance allowed only inside the boundaries of the defined protocol."

So instead of some arbitrary language on a website, the warrant canary standard would be a well-defined JSON - that's J-S-O-N JSON, not J-A-S-O-N Jason.

JASON: It gets me every time.

Steve: I know. Your ears perk up - a JSON-formatted text file which is convenient since it's readable both by people and by machines. The CanaryTail file would include 11 codes with a value of zero for false or one for true. These codes would include things like W-A-R for warrants, G-A-G for gag orders, and T-R-A-P for trap and trace orders, along with another code for subpoenas and so forth, all having specific legal meanings for an organization and its users. Naturally, a value of zero adjacent to any of these keys would mean that none of the warnings had been triggered. If the code changes to one, it's a cause for concern.

And of course the beauty of this is presumably that these files would have a common name, would be in the root directory of the domain hosting the canary, and this would allow for automated verification. And if this really did take off as a standard, you can imagine, for example, our web browsers might be - or certainly an extension. I guarantee you that there would be an extension for our web browsers which would poke at the canary and bring up some sort of a warning with an icon jumping up and down if a non-zero item was found on one of these files. So it's a cool idea.

The file also contains some interesting codes including DURESS, which would indicate that the organization is somehow, in some way, being coerced; along with codes

indicating that they've been raided. There's also a special code - I love this - indicating a seppuku pledge. I first encountered the term on "Serenity," that wonderful movie that followed the "Firefly" series. Seppuku, as in the ancient Japanese suicide ritual. In this case the seppuku pledge that would be contained in the warrant canary file would be a pledge that an organization will shut down and destroy all of its data if a malicious entity were to somehow take control of it. In other words, it would kill itself before it would allow itself to be abused, and the knowledge, the information that it contains.

And of course because these days you can't have anything good without somehow tying in Bitcoin's blockchain into the whole thing, the proposed standard must also be cryptographically signed, including a public key, with an expiration date, and must include a block hash from the Bitcoin blockchain to verify the freshness of the digital signature. As another safeguard, it would include a panic key field with another public key. If the file is signed with that key, people could interpret that as a kill switch, causing the warrant canary to fail immediately. That would be useful if an organization were to suddenly get raided and can't afford to wait until the current canary file expires. So anyway, a lot of thought obviously put into this idea. And who knows? We may end up with a standardized warrant canary which our browsers could be checking for us whenever we go to a website that has one, which might be kind of cool.

JASON: So if our browsers are checking for it, then I would imagine - so let's say an organization has to switch one of these flags so that people kind of know if something has happened, law enforcement or whatever the cause is of that. They know, as well; right? Like the public knows because the browser is identifying this. But it's no secret to them, either; right?

Steve: Right.

JASON: They're going to know this, as well. So, I mean, is there - I guess I don't know what my question is. Pardon me, I'm a little foggy. I've got a cold. But, I mean, that kind of takes some of the secrecy out of it. I guess when I've always thought of the canary, the warrant canary, I've always thought of this, like, this mysterious hidden kind of signal to everyone else that's in the know that, oh, we know how to crack the code. We now know that there's something that's happening to this site or this organization that maybe they don't know that we know.

Steve: Yeah.

JASON: Does that make sense?

Steve: So, yeah, it does. And I think where your previous understanding differs from the way it's been applied is that the canary is useful because people have a way of discovering it. That is, maybe in their privacy statement they will say, oh, and just as a verification, keep an eye on our warrant canary. I mean, so the idea that a site would be hosting a warrant canary, that itself is not secret. And so, for example, that's why when SpiderOak changed theirs from explicitly calling it "here's our warrant canary" to "here's our transparency report," people all freaked out. Well, of course they were only able to freak out because they knew that it had been previously called the SpiderOak warrant canary.

So the idea was that it was supposed to be something that was discoverable by anybody who was interested, and it would be up to them to keep an eye on it. Or like, for example, maybe before they would go back to update their software with a new version they might, if they were of the tinfoil hat-wearing ilk, go see if the site has a warrant canary attesting to the fact sort of passively that, yup, you know, everything is still fine. Nothing to see here. Move along. Anyway, so it was never really meant, you know, they

weren't meant to be secret. They were meant to be discoverable by somebody who wanted to poke around in their privacy statements and see what they could find. Or be also passed from person to person. Like hey, you know, make sure you check out the site's warrant canary before you trust them overly.

JASON: Right, yeah. Okay, that makes more sense.

Steve: So this is one I'm glad...

JASON: I love this one. I'm so happy you put this in here, thank you.

Steve: I'm glad you knew of it and you were hoping that we would talk about it. This puts the "wow" in Xiaomi.

JASON: Yes, it does.

Steve: As we know, recently a number of, well, what we're finding out is that a number of users of Xiaomi's smart, or now it seems apparently not so smart, IP-connected IoT cameras - get a load of this - have been receiving and watching other Xiaomi users' video feeds. Whoops.

JASON: Ouch.

Steve: For whatever reason, what is clearly an unintended bug appears to only affect Xiaomi IP cameras when they're 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, M-I-J-I-A, 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, unbidden. So the problem does not appear to lie in anything that Google did in its implementation, but rather is something about the way Xiaomi implemented their connection between their smart cameras and Google's Nest Hub.

In a statement Google made which was provided to the publication "Android Police," Google said it's aware of the issue and is in contact with Xiaomi to fix the problem, although Xiaomi had not yet responded to Google, which I think was a mistake because, in the meantime, as a precaution to protect its Nest Hub users, Google has temporarily disabled all Xiaomi devices that are attempting to get to the Internet through the Google Nest Hub. So that will certainly get the attention of Xiaomi's users and no doubt get the attention of Xiaomi, as well. The exact details surrounding the bug are still unknown, but it does not appear to be affecting all Xiaomi cameras which have been integrated with the Google Nest Hub. So it's a mystery. Who knows what the story is.

I would suggest and recommend that, until patches arrive, that users unlink their cameras from Google Nest Hub - that is, Xiaomi users, only Xiaomi users, unlink their Xiaomi IP cameras from Google Nest Hub, if that's the thing that you're using to tie them to the Internet - and wait, as I said, for patches. And of course the takeaway for our podcast is always to keep in mind that in the past, in 2019, and moving forward into 2020, any connection to the cloud truly does remain a high-risk business. It just is. Everything that we see inherently suggests that.

Most organizations want to offload the burden of using the cloud from their users, which is why, for example, Dropbox and Microsoft, while they encrypt their customers' data at rest, they have the keys. Dropbox, their terms of service specifically say "Our employees may need to look at the contents of what you're storing in your Dropbox in order to verify it meets the terms of service." So they can obviously see it. So we're still in the

early stages, it feels to me. Perhaps we will eventually, as a society, as an IT environment, we will eventually learn how to secure the cloud in a way that's useful.

But we're about to talk about Microsoft's Remote Desktop Protocol and find that they still haven't figured it out, which is amazing to me. 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 even now. So whether you're outsourcing your management functions to a third party who then becomes a conduit to carry destructive ransomware into and throughout your organization, or you're using cloud-connected surveillance cameras and lord only knows who can also see whatever the cameras are aimed at, there's just no arguing that, while there are absolutely significant upside benefits, so far they don't come without some significantly balancing downside risk.

So at this point it's the Wild West, and the technology does cool stuff. You know, it's neat that you're able to look out from your front door of your home while you're at the office. There are secure ways to do that, that don't involve the cloud, that would arrange to only allow you and your office to see your video. But it takes a lot more configuration; and it's not the "oh, look, I just plugged it in, and suddenly it all works" experience that users are used to now. The flipside of that is, yeah, other people may be getting your video because of a bug. So, whoops.

JASON: And, I mean, also important to point out this was a small number of users. Xiaomi, I think, or someone revealed it was about a thousand cameras or something like that, or a thousand users. But, I mean, that number is kind of pointless in the perspective of people who are considering bringing smart technology into their home, but are worried about whatever the inherent risk is. And when you're talking about bringing cameras into your home, of course one of the biggest worries is how do I know that the right person - i.e., me or my family - is going to see that video feed and no one else is, especially depending on where you put these security cameras. This just ends up being like the perfect storm example of, like, oh, shoot, like that's a worst-case scenario, and yet some random person had this other random person's video feed from inside their home. That could be really bad. It's not good.

Steve: Yeah, and if anyone has been receiving that spam recently that talks about seeing you doing something that you don't want anyone to see you doing, so pay them some bitcoin or they're going to release it publicly, well, this puts a little more veracity. I mean, that's just nonsense. That's not happening. No one should believe that spam. But it's like, ooh, you see this in the news, and you can imagine if the local news picks up a little blurb that, gee, the wrong people are seeing other people's video feeds. It's like, what? What? That's not good.

JASON: It's perfect, perfect fodder.

Steve: So, yes. So you make the point, I mean, it is, yes, it's really cool and neat, and we would wish that mistakes were not being made. But mistakes are being made.

JASON: There's humans involved here. We aren't going to be able to create 100% perfect technology. As for Xiaomi, their devices are back online except for this one particular camera. So apparently Xiaomi has isolated it to this camera, and the services there are not active yet. So I don't know when that'll happen. Obviously Xiaomi's looking into it. But, yeah.

Steve: Well, so here we have now our next case of what is not a mistake. And that's just what - this is the part that boggles my mind. This is not a mistake. This is the way Microsoft designed it. On December 18th they posted, the security group posted a blog titled - very high-falutin' title - "Data science for cybersecurity: A probabilistic time series

model for detecting RDP inbound brute force attacks." And I have a link in the show notes for anyone who wants to glaze over more than that title already did. But I'll just share the introduction. It's a long, long posting. But you can see what the intent was from the introduction.

They wrote: "Computers with 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." Yeah, no kidding. "Brute forcing RDP" - then Microsoft says, and I got a kick out of this, too - "a secure network communications protocol that provides remote access over port 3389" - okay, well, stop right there. It isn't a secure network access protocol. Last year they were forced to go back and patch Windows XP because it was not secure. And half a million Windows machines right now still don't have that patch applied. So they would like it to be, they wish it were a secure network communications protocol. And I would argue that so long as you have a brute-forcible authentication that doesn't keep people from brute forcing it, it can't be called a secure network communications protocol. But okay.

They go on. They say: "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." Right, because it's not secure. Instead, they said: "Attackers can utilize many off-the-shelf tools" - many - "to scan the Internet for potential victims and leverage similar such tools for conducting the brute force attack." So off the shelf, wonderful.

"Attackers," they write, "target RDP servers that use weak passwords and are without multifactor authentication" - because it doesn't offer that - "virtual private networks, 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."

They said: "In a brute force attack, adversaries attempt to sign into 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." What do you know? Imagine that. They said: "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 [Microsoft] 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" - which actually means success - "for non-server Windows machines can shed light on which sign-in sessions were successfully created and can help further in detecting" - yeah, after the fact - "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" - they're not fixing the protocol, but they're going to analyze its failure. "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." Again, okay, so we're not going to fix this. We're just going to analyze it.

So here's what we know now as a consequence of this report - I read the rest of it so that we don't have to on the podcast - that we didn't know before. Microsoft learned that approximately 0.08% of RDP brute force attacks are successful and that brute force attacks last two to three days on average. For the study, Microsoft collected data on RDP login-related events from more than 45,000 workstations running Microsoft's Defender Advanced Threat Protection. And this is the instrumentation that we are always talking about, these machines phoning home and providing useful information to Microsoft. Anyway, that Advanced Threat Protection is the commercial version of the free built-in Windows Defender AV 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 the IDs I mentioned before, 4265 and 4264, which respectively provide the usernames of a user or an attacker based on whether they succeeded or failed. As we know, RDP is often deployed in enterprise environments to allow Windows sysadmins to manage servers and workstations in remote locations and can also be used by employees to access their machines when they're away traveling.

So we also know - horrible authentication bypass mistakes in RDP notwithstanding - that attackers 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 in an attempt to guess the target computer's RDP login credentials. These attacks typically use combinations of usernames and passwords, maybe that have been leaked online after breaches of online services. And we know that massive lists of usernames and passwords are easily obtainable these days. They are available for easy download, and they're often used.

Microsoft says that their recently observed RDP brute force attacks, as I mentioned, typically last two to three 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 are deliberately throttling their access in an attempt to avoid having their IPs detected as malicious and therefore banned by defensive third-party firewalls. Consequently, rather than blasting an open RDP port with thousands of failing attempts at once, attackers try only a few combinations to 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," they wrote, "we found that about .08% were compromised." So I did the math; and, at a success rate of .08%, that's 36 machines out of 45,000. So, okay. That's not many. But we're talking about a bad guy remotely getting into your system and being able to sit there just trickling login attempts over some length of time. And of course we know that the use of botnets expands the rate at which brute forcing can be done because a botnet can be used in order to provide a mass of IPs so that a single IP cannot be blacklisted and shut down all attempts to get in.

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

And I don't know. As our listeners know, I manage myself a number of Windows servers remotely. I find the use of Remote Desktop Protocol very convenient. But you won't find a single instance of RDP listening on any publicly exposed port within any of GRC's IPs. It should now be clear as day to anybody that there is no safe way to have RDP publicly exposed ever. You know? Tuck it safely behind a VPN with multifactor login, including a

time-based token and a client credential certificate required, and then the entire problem just goes away.

But instead, perversely, Microsoft concludes their otherwise useful and interesting research report by recommending that systems admins combine and monitor multiple signals for detecting inbound RDP brute force traffic on a machine. In their report they finished, they concluded that the signals that you should watch were hour of day and day of week of failed sign-in of RDP connections; timing of successful sign-in following failed attempts; Event ID 4625 login type, filtered to network and remote interactive logins; Event ID 4625 failure reason, filtered to include subtypes 2308, 2312, and 2313; cumulative count of distinct usernames 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.

I mean, again, they're trying to fix something that's completely broken. And I was trying to think of an analogy. It would be like saying to the vendor of a truly impenetrable security door, "Oh, I know that your doors are impenetrable and inexpensive, and that the door we have here is kind of fragile, flimsy, and falling apart. But we've decided that 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." And then when the vendor says, "What receptionist?" he says, "Oh, yeah, you didn't see our receptionist when you came in? No, she doesn't always feel well on Mondays."

So again, here's all of this effort being focused on how to determine if your publicly exposed, fundamentally insecure, and insecurable Remote Desktop Protocol server is being brute forced. Okay, first of all, Shodan lists them all. I mean, you can get a list of them all. So, yes, your publicly exposed RDP server is going to be brute forced. So what good does watching that do? Fix the problem. Anyway.

JASON: Well, at least we know. At least we saw it happen.

Steve: Yeah, yeah. I guess that would be justification for buying the really good impenetrable door.

JASON: Yeah, sure.

Steve: Or what's the problem with a VPN? They're free. Get one. They don't cost anything. Put a pfSense box for 150 bucks in front of it. I mean, it's like, okay, problem solved. Amazing.

JASON: All right. So what do we have next? Point of sale.

Steve: Well, here we are at the beginning of 2020, and point-of-sale systems continue to be a huge problem. They're one of the earliest problems, and we just don't seem able to solve these problems. The hacks, I guess, are maybe less gee-whiz, a little less technically interesting. And of course credit card theft typically represents a widely distributed, though individually recoverable pain, as I well know, actually, having had more than my share of credit cards canceled and reissued after incidents of online fraud. I've told the story a number of times of back when I was using a travel agent. Every year Judy would say, "Okay, Steve, are we changing our card again?" It's like, yes, Judy. That would happen because I got a fraudulent charge back in those days. Now I'm able to aggregate a lot more of my purchases through a few vendors, or PayPal, if sites allow a purchase through PayPal. So the problem seems less.

But point-of-sale terminals continue to be a problem. In our Decade of Hacks retrospective two weeks ago, Leo and I remembered the history-making attack on the

Target chain of retail stores. That was a huge point-of-sale attack that really sort of put the whole idea on people's radar. So I just wanted to remind everyone as a consequence of a couple events that have just happened, that these attacks have not stopped or even really slowed down. In this case another major restaurant chain - or chain owner, actually, Landry's - recently revealed a longstanding attack on its POS, point-of-sale systems.

When I heard "Landry's," I didn't think much of it for myself since I've never eaten at a Landry's restaurant, or so I thought. When I looked over the chains that they own, however, and operate, I recognized many of them. 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 under the Landry's umbrella. They do have a seafood restaurant. There's the Chart House, Saltgrass Steak House, Claim Jumper, Morton's Steakhouse, Mastro's Restaurants, Rainforest Caf, just, I mean, and this list goes on and on. So it's likely that, if you are a diner, that you've eaten at one of these places.

So according to their own breach notification which they published this week, the malware was designed to search for and steal sensitive customer credit card data. It's funny, too, because when I read this, "malware is designed to search for and steal," when we're talking about our lexicon, we'll be talking about data scraping malware that does exactly that. It lives to search out and find stuff that it's been targeted for. Anyway, credit card numbers, expiration dates, verification codes and, in some cases, cardholder names.

This point-of-sale malware infected point-of-sale terminals at all Landry's-owned locations. So it was pervasive. They said, and this puzzled the tech press that was covering this, that the end-to-end encryption technology used by their terminals prevented attackers from stealing payment card data from card swipes at its restaurants. But then they said that their outlets also use "order entry systems with a card reader attached for wait staff to enter kitchen and bar orders and to swipe Landry's Select Club reward cards." So apparently somehow that allowed the wait staff to mistakenly swipe payment cards, which were not end-to-end encrypted, and allowed the data to escape.

So no one's really clear how many, what it meant. Landry's not being forthcoming. They did, however, say that this malware was actively scanning their systems between the 13th of March in 2019 and the 17th of October in 2019. So for, what, more than half a year this stuff was in at all of their locations. And they also said it may have been installed as early as the 18th of January. So for what it's worth, they're clearly trying to at least disclose the nature of the problem. And they said that during the investigation they had removed the malware and implemented enhanced security measures and are providing additional training to wait staff. So maybe it was a human factors problem, as well. Who knows?

Also, however, another major U.S. convenience store chain, Wawa, W-A-W-A, also said last month that it had recently discovered malware that was skimming customers' payment card data at just about all of its 850 stores. In this case the infection began rolling out to the stores' payment card processing system on March 4th and was not discovered until December 10th. So again, nine months of active slurping up of anyone who used a credit card or debit card, I guess, at Wawa. They said it took, after the discovery, two more days for the ransomware to be fully contained. Most locations' point-of-sale systems were infected by April 22nd, apparently, although the advisory said that some locations may not have been affected at all.

And they said that the malware collected payment card numbers, expiration dates, cardholder names from payment cards used at potentially all Wawa in-store payment terminals and fuel dispensers. It did not disclose how many customers or cards were affected. They said the malware did not access debit card PINs, credit card CVV numbers,

you know, the little authentication numbers on the back of our cards, or driver's license data, which might have been used to verify age-restricted purchases, you know, presumably alcohol. They said information processed by in-store ATMs was also not affected, so that was clearly an independent network. They said they'd hired an independent forensics firm to investigate the infection.

So anyway, I just wanted to mention that point-of-sale problems are not gone. They tend to be, you know, they hit the news when they're huge. But they're generally, I mean, for example, I'm seeing them and not bothering to bring our podcast listeners' attention to them constantly because again, as I said, they're sort of low tech. They're not that interesting. They tend to be regional when they're a small local chain, so they're not affecting everybody. So it probably hits the local news.

And in general everyone using a credit card online would be well advised to scan their statement. Certainly if huge payments appear, that will tend to stick out like a sore thumb. But a lot of these things are deliberately small dings against a credit card, specifically to keep it under the radar. When these cards are being aggregated, they're then sold off in the black market to bad guys for whom these cards are very valuable. So anyway, certainly keep an eye on, you know, just scan through and make sure that you recognize all the payments on your card. It just makes sense to do that.

I titled this next piece "Be careful when you flip that switch" because, according to the Russian government, their recent Runet disconnection test was successful. Monday of Christmas week the Russian government announced that it had concluded a series of tests during which it successfully disconnected the entire 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, which we've talked about previously, was to determine and verify whether the country's entire national Internet infrastructure, known inside Russia as Runet, could function without access to the global DNS system and the external Internet. And of course this disconnect, especially DNS, which immediately came to everyone's attention as, oh, how are you going to do that, you know, dropping off the Internet is not easy. After they pulled the switch, all Internet traffic was rerouted internally, effectively making Russia's Runet the world's largest Intranet, that is, a big network that is not connected to the rest of the Internet, although Intranets normally have gateways that hook them to the Internet. It became a private network.

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 cyberattack from a foreign country. I'm not sure that would work that well, but what the heck. A cyberattack would normally have some sort of technical agency within the environment that it's bound to attack, and it would probably work autonomously, especially now that Russia has just announced to the whole world that, hey, we're able to pull the switch. So if you really did want to attack Russia, you'd arrange to be switch-resistant. And we know there are lots of ways to do that.

But in any event, this was all cited by multiple Russian news agencies. And Alexei Sokolov, who's the deputy head of the Ministry of Digital Development, Communications, and Mass Media, said: "It turned out that, in general, 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." He said the results of the tests would be presented to President Putin in 2020. And of course we've discussed, as I mentioned, this plan several times in the past. So we already understand its outline.

The tests were the result of many years of planning and included some required lawmaking by the Russian government, as well as physical modifications to Russia's internal Internet infrastructure. The tests had originally been scheduled for April of 2019, but ended up being delayed until this week of before and during Christmas to give the Kremlin time to pass their new so-called "Internet Sovereignty Law" which grants the Russian government the power to disconnect the country from the rest of the Internet at will, which it apparently didn't have before that, and with little explanation, on the grounds of - yes, you can just imagine the next phrase - "national security."

So to enable this the law mandates that all local Internet service providers be able to reroute all Internet traffic through strategic chokepoints under the management of Russia's Communications Ministry. So as I understand it, basically there's been a big rewiring which has occurred in order to make this possible. And ISPs are now running their traffic through these chokepoints. And these chokepoints thus serve as a gigantic kill switch for Russia's connection to the external Internet. And it's been noted that it also provides a powerful opportunity for Internet surveillance, not unlike China's Great Firewall technology. That is, when you've got all Internet traffic transiting a few chokepoints, well, that's where you put your big monitoring technology if you want to see what's going on through those chokepoints.

So as I mentioned, it's truly difficult in this day and age to consider - I can't imagine having the U.S. being disconnected from the entire rest of the world. I mean, who knows what the consequences would be, what things, what critical infrastructure things would break when they lost contact with things outside of our geographic boundary? It's a little bit like the Y2K problem.

JASON: Exactly.

Steve: It really is something that you need to consider very carefully. And frankly, I have to say I'm a little envious of Russia that they have the ability to do this. It's a challenge, and it did take them, even with as much control as their government has, it took them years to be able to pull this off from just a technology standpoint.

JASON: I don't know. I can't imagine much harm would happen if the U.S. was cut off from the rest of the world. I mean, I'm an optimist. So maybe that's my flaw. But I don't see how anything could possibly go wrong.

Steve: Okay.

JASON: Steve's like, "There, there. There, there."

Steve: Okay, Jason, yeah. Speaking of things not going wrong, Python 2.7 has reached its end of life after 20 years. How quickly 20 years passes. Python 2.7 was released in the year 2000. Wow, you know, like back when Windows 2000 server was like the thing. Wow. So Python 3 released six years later than 2.7, in '06. But as we so often see with technology, if it's not broke, leave it alone. So Python 2.7 was working quite nicely, thank you very much. So why would we want to move to 3? Even though now we've had 3 for, what, 14 years. And sure enough, due to 2.7's incredible popularity, even though it had its original sunset retirement planned for 2015, the Python developers decided, oh, what the heck, we will continue supporting both development branches.

But I'm sure it hasn't escaped anyone's attention by now that it is, as we've said already, year 2020. And yes, after a good 20-year run, Python 2.7's retirement day finally arrived six days ago. Python 2.7 was officially at end of life at the beginning of this year. 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 all of their attention on Python 3 with an eye toward

increasing its speed and patching any bugs and just doing a better job, not having to continually split and back port things that they're doing and caring about on 3 back to 2.7, only because everybody else is using it. So they finally said, okay, enough already. We're done.

In their announcement they wrote: "We are volunteers who make and care for the Python programming language. We have decided that January 1, 2020 will be the day that we" - well, and in fact it turned out not to be, but we'll get there in a second - "will be the day that we sunset Python 2. That means we will not provide any more after that day, even if someone finds a security problem in it. You should upgrade to Python 3 as soon as you can."

Now, unfortunately, having said that, the team does plan one last release of Python 2.7 in April. That final release will include any final bug and security fixes that may have been hanging over from 2019 and, frankly, any more up until that final release date that they have the opportunity to fit in in time. They said: "We want to leave Python as solid and stable as possible."

But it's clearly time to move on. Bleeping Computer noted that for those who really do still require, and will going forward, Python 2.7 compatibility, and for whom for whatever reason moving to 3 is a problem, an option would be to move from Python 2.7 to PyPy. That's P-Y-P-Y. It's an alternative reimplementation 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, or when compiled under PyPy, than interpreted under Python. Also PyPy-compiled programs also tend to require less memory to execute. So it's worth considering.

On the other hand, 2 is going away. Most Linux distros are working right now to migrate their dependencies and their libraries from 2.7 to 3. Debian, Ubuntu, and Fedora have begun the process of updating Python 2 libraries to their Python 3 equivalents. Debian's so-called "Buster," Debian's Buster 10x release and Ubuntu 18.04 LTS will both be using Python 3 as their default, but Python 2.7 will still remain available for those wishing to install it deliberately. 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 has retired Python 2.7, they will continue to support it through the normal RHEL lifecycle, so that continues it until 2024, so for another four years. So anyway, if you are a Python user, you need to consider 2.7 and 3 and see if you're able to move away from 2.7 because it's going to be getting kind of scarce.

It's fun to check in on HackerOne. As our listeners know, we're quite bullish about the emergence of this new category of employment, security vulnerability hunting and reporting for profit. This part- or full-time career is enabled by companies such as HackerOne, my personal favorite, since unlike Zerodium, HackerOne does not hoard and then turn around and secretly sell their submissions into the gray market for potentially malign use by national law enforcement, intelligence, and cybercrime entities. Rather, 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. And so that's what they're doing. So it's instructive to check in every so often to see what the top 20 HackerOne clients are up to.

I have a link in the show notes to the extensive PDF covering a complete detailing of it. I'll spare our listeners that. But to give you a sense for it, ranked downward from the most money paid down, the top 20 companies are Verizon Media in the number one spot, then Uber, PayPal, Shopify, Twitter, Intel, Airbnb, Ubiquiti Networks, Valve, GitLab,

GitHub, Slack, Starbucks, Mail.ru, Grab, Coinbase, Snapchat, HackerOne themselves - that's in the 18th place - Dropbox in 19th, and VK, which is a social media domain, are the top 20.

Verizon's spot is solidly held. They have paid out more than $4 million since launching their program back in February of 2014. So they are number one in all-time bounties paid and in the most hackers thanked and in the most reports resolved. Number one in all three. Now, part of that is early participation. 2014 compared to some of these other newcomers is a long time. So they've had time to accrue a high total bounty. As noted by number two, that's Uber, who has less than, but not much less, than half the total bounties paid at 1.8 million, whereas remember Verizon is at 4 million. So Uber is at half that, or a little less than half that, at 1.8.

But whereas Verizon's top bounty, top single bounty ever paid was $6,000, Uber has paid one of $15,000. And also Verizon's 4 million are spread across a total of 5,269 reports, whereas Uber's only slightly less than half, 1.8 million, is only spread over 1,172 reports. So it does appear that Uber is paying bigger bounties on average. And I don't know if, like for example in Verizon's case, maybe Verizon's more recent bounties are greater because there's been an inflation in and competition in bounty payout such that in the same more recent timeframe it might be that Verizon is paying more expensive bounties to be more at parity to what Uber is. We really can't tell from the data.

PayPal, in the number three slot, takes the award for the single largest bounty paid through HackerOne, $30,000. And they're also clearly paying more per bounty since their all-time total payout is one bounty of $1.25 million. Oh, no, wait. No, no, no, no. I got that wrong, sorry. They're clearly paying more - right. I'm sorry. The largest single bounty I just said was $30,000. But they're also clearly paying more per bounty since their all-time total payout is $1.25 million, spread across only 430 reports. So that's by far a larger per-report bounty. And they've only been a participant in HackerOne since 2018. So despite that, they are in the top five in fastest response time, that is, in terms of paying bounties after the reports. So they clearly see this as an important and significant security win for them.

So anyway, I'm not going to keep enumerating every detail of the next 17. Again, all of those details are available through the link in the show notes. But I will note that Shopify boasted the fastest time to bounty payout of only two days and is among the top five in most reports resolved and the largest single bounty paid. Valve is also among the top five paying the single largest bounty, and it has company with GitHub, Coinbase, Snapchat, and 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. And, you know, if you're still living at home, and you're a hacker, maybe you can hack your way into your own space and give your Mom and Dad a break.

JASON: Right. No, I promise, Mom and Dad. My hacking is for good. By the way, here's a multiple thousand, $30,000 payout. That's not bad. But there's probably tons and tons of people going after this. So I don't know. Your odds, I'm not sure what that...

Steve: There is certainly competition, and it is a function of being good. But a lot of people think they're good.

JASON: Everybody.

Steve: And this could be - yes, exactly. And this could also be something where, as they say, "Don't quit your day job."

JASON: Right.

Steve: But which you literally, rather than sitting mesmerized in front of a ridiculous videogame, like all evening every evening, get creative. Sharpen up your hacking skills. And if you find that you can start earning some bounties, then maybe your employer needs to give you a raise.

JASON: Not bad. Not bad at all.

Steve: We'll finish up before our last break and getting into our fun stuff with this week's possible action item for some of our listeners. As we often do, we have one potentially significant and widespread new vulnerability that we need to inform our listeners of. The prolific Tencent Blade Team, that's the Chinese hacking team that is very good, they recently disclosed another batch of serious SQLite (S-Q-L-I-T-E) vulnerabilities which they have named Magellan 2.0. These newly discovered vulnerabilities occur in obviously the extremely popular and widespread SQLite database, which I see installed everywhere. I mean, it's sort of an embedded application-scale database which allows an app to store data locally for whatever purpose and spares the app developers the burden of doing it themselves.

So it's a very compact, relational database management system that's used, I mean, I see it all over the place. It is, significantly, in Google Chrome, Mozilla Firefox, Windows 10, and all over the place. It's been nearly a year since the Tencent Blade Team disclosed their original batch, which they called Magellan 1.0 SQLite vulnerabilities. These new revelations affect all programs - get this - all programs that utilize SQLite as a component in their software. And the vulnerability manifests if they allow external SQL queries. Now, that's the good news. It is probably a tiny percentage of apps where their use of SQLite also exposes SQLite to the public. Most of the library use is just internal database purposes like, well, you can imagine. I would be surprised if Thunderbird, the email client that I use, doesn't keep all of its stuff in SQLite. That's just what you do these days.

What's significant, and Chrome has already fixed this, is that one of the biggest targets was Google's Chrome browser. What they found was that, as a consequence of the way Chrome operated, if the browser had Web SQL enabled, it was possible to go through a web page and get to SQLite and turn this into a remotely exploitable remote code execution vulnerability. They said: "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 software, so this vulnerability has a wide range of influence. SQLite and Google had confirmed and fixed these vulnerabilities. They were disclosed responsibly." They said: "We will not disclose any details of the vulnerability at this time, and we are pushing other vendors to fix this vulnerability as quickly as possible."

So anyway, using these vulnerabilities, Tencent was able to remotely execute commands in Google Chrome as long as, as I mentioned, this Web SQL was enabled in the browser. So this critical vulnerability was closed by Google. The SQLite team were notified of this round as they were almost a year ago of the Magellan 1.0 problems. SQLite was patched for these only on the 13th of December, so not even a month ago yet. So, and of course it itself does support remote queries. If anything you're using, like for example is using UPnP to map a port open so that some component of something that is SQLite based has public access, that would mean you are in serious danger. So I wanted to bring it to everyone's attention.

There were five CVEs issued. They've all been fixed and patched. But of course it doesn't do any good unless SQLite itself gets patched. The other problem is there could well be

ways of locally exploiting SQLite to execute code which malware might start being clever about using. As I said, the problem is it is so widely used that, when you discover a problem like this, it's going to be everywhere. And because it's an embedded component of something you may have had in your machine for who knows how long, that may not update itself, and you certainly wouldn't even be aware that SQLite was in there as an embedded component, it really does represent a problem potentially moving forward. 2020 may be seeing some SQLite-related exploits going forward. It wouldn't surprise me.

JASON: All right, got some sci-fi right here. I like this.

Steve: So I tweeted out to all of my Twitter followers when I was clued in. It wasn't until, well, sometime between Christmas and New Year's that Netflix had dropped the second 10-episode season of "Lost in Space." And that's now behind me. And I enjoyed it, although I have to say that Dr. Smith, oh, she is deliberately designed to be so annoying, and she's fulfilling her role well. Also "The Mandalorian," I finally caught up and finished all of that. And so I will be resigning from Disney because there's nothing else that I want to watch there except, well, I've seen all the Star Wars stuff.

JASON: So you signed up for the trial? Or did you do like a month? Because, I mean, "The Mandalorian" has been going now since it launched, I guess what is that, a month and a half?

Steve: Yeah. I did sign up. I thought it's only fair to do the subscription. And it goes, for me, it goes through the 24th. So I will resign, you know, it's $7 a month, so it's not a big deal. But again, I think that's sort of generally what people are doing now is they sign up for something that has a specific thing they're interested in watching, they watch it, and then they go, okay, well, there's nothing else here. I'll come back next year for the next installment.

JASON: Totally. Yeah, I'll be curious to hear because I wasn't aware that "The Mandalorian," the releases were wrapping up. Like I've heard nothing but great things about it. At some point I'll watch it, along with everything else that I've said that about. But I would be really curious to know with Disney Plus how many people do exactly what you did; right? Like we signed up because we knew "The Mandalorian" was a thing that we wanted to see. And Disney's super happy that everybody did that. And then what is going to be the rate of people dropping it and pausing it until the next season?

Steve: Well, and sadly, I think what the outcome of this will be, if this becomes like a thing that people do, is they'll, I mean, nothing requires them to drop them all at once and allow people to binge on them in one week and then resign. They could certainly dribble this out once a week. And, well, in fact, no, wait, "The Mandalorian" was dribbled out once a week.

JASON: Oh, yeah, that's how they did it, yeah.

Steve: Yeah, it was "Lost in Space" where it was all available from Netflix. And thank you, Netflix. I'm keeping my subscription because there's lots of other goodies there that I want to watch.

JASON: Sure.

Steve: So I think they're probably safe from that. But so I see what you mean. Yes. Once a week for 10 weeks, and now goodbye, Disney.

JASON: Right. Until the next "Mandalorian."

Steve: Yeah.

JASON: Yeah, it's interesting that they both, between Netflix and Disney Plus, they have the different release approach. And I'm not sure which is better for...

Steve: Yeah, and I think if they tried to stretch it out over a year, like if they did one a month, that's not going to work, either.

JASON: Right. I don't think that would work.

Steve: People would go, "Screw this. I'm not, you know, I'm not going to wait for this."

JASON: Absolutely. Or you'd wait for the year and then sign up and watch and then binge on them, which people could totally do after the 10 weeks of episodes of "Mandalorian."

Steve: So I do have a neat announcement. As SQRL continues to spread, we have now SQRL support, native SQRL support for Drupal. A contributor of ours, I guess I pronounce his name Jrgen?

JASON: Jrgen.

Steve: Jrgen Haas. He wrote: "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." Then he has links to a test server. He says the source code with the Drupal module is available from, and we have a link to that. He said: "The installation on any existing Drupal 8 site is as simple as" - and then he has the command - "composer require drupal/sqrl," followed by, and then it's like "drush en sqrl," D-R-U-S-H E-N S-Q-R-L, he says, and that's all to get it up and running. And then he said: "Maybe a new forum section on sqrl.grc.com might be worth having." And he says: "To me this is a great start into 2020, and I wish everyone around a Happy New Year."

Anyway, so I thanked Jrgen for his terrific work, told him that I'd be mentioning this, as here I am on this week's Security Now!, and that I would immediately set up a forum at sqrl.grc.com for him to moderate for the public management of this great work. And that forum has been in existence now ever since then, like a week plus. And he's there, and a bunch of people are hanging out and playing with it. And in fact shortly after that a Brian of London chimed in, posting, he said: "And as one of the first testers, I can confirm it was ridiculously easy to fire up a SQRL client," he said, parens, "(can't even remember which client I used), sign in for the first time, and then set up an avatar and so on." He says: "I haven't given it an email or password. None needed." So very cool continuing progress on the SQRL front.

And as I mentioned last time, when I formally turned the SQRL project over to its terrifically capable community, this past holiday period was my work time, well, actually it was time I was giving myself to work on some long-needed attention to GRC's servers at Level 3. So that has all happened. I'm not quite finished yet. Our long-running FreeBSD v5 Unix DNS and news server hardware I finally replaced after 15 years of flawless nonstop service. There's FreeBSD Unix for you. And nothing was wrong with it. It was working just fine, even after all that time. But I wanted to move us from a 32-bit architecture to 64.

We're now running FreeBSD 12.1, which is the latest. That allowed me to update to a current version of BIND. We're running BIND 9 for serving DNS. And, you know, new versions of OpenSSL and so forth. So anyway, I'm still wrapping up the re-addition of some of the deep and useful customizations that I had made to the old news server code.

You know, it's all written in C, and I speak C. So there are a number of - we have a number of enhancements to the standard news server that provides everyone who's over there a bunch of special and useful features. So I'm in the process of reimplementing them from 15 years ago. And then I'm just getting, you know, sort of clearing the decks to get back to SpinRite 6.1, which is coming soon, I'm happy to say.

JASON: Nice. You've got a lot going on.

Steve: Never a dull moment.

JASON: Never a dull moment in the '20s.

Steve: Yes. Our Malware Lexicon. So, you know, we're always talking about this or that sort of threat. And just like with biological viruses, malware evolves over time. Some malware is highly opportunistic. It rises briefly to capture our attention while it capitalizes on a short-term opportunity. Then, when that opportunity disappears, so does it. But many others have evolved to exploit more fundamental flaws and problems that we have not yet as an industry managed to resolve. For example, the ongoing troubles with point-of-sale systems that we were just talking about earlier are a good example of an early problem that persists. And then of course the new and clearly justified worries over the security of IoT are a good example of a relatively recent new threat. So some things are old, and they're enduring; other things are hanging around.

So I ran across a nice summary breakdown of sort of a malware methodology by Sophos. I thought it was an appropriate way to ring in this New Year to sort of quickly run through the lexicon of the terminology that we now use to describe the various sorts of enduring threats which continue to evolve and mature. They're the Deadly Seven. There are seven of them.

Number one, keyloggers. They're surprisingly simple and can be implemented in many different ways. They hook into the stream of data coming from our keyboards which allow them to capture everything we type. As anyone who has ever seen one of my SQRL presentations knows, I often use the example of being in Las Vegas and needing to log into my Southwest Airlines account to get a boarding pass 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. And the reason I use it in a SQRL presentation is that SQRL of course allows for, it provides a zero keystroke login to untrusted PCs. From the PC's standpoint, and from the standpoint of any malware that is in it, I'm just suddenly logged in without doing anything. So there's no keystrokes for anybody to capture.

And of course the keyloggers, as a consequence of the way our keyboards work, keyboards actually transmit a key cap down/key cap up individual messages. So they don't merely know that we type F. They get enough detail to tell that the user pressed the left Shift key down, then pressed F, then released F, then let go of the Shift key. So 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 might be used to turn, you know, Alt and Control things that turn options on and off.

And of course, as we also know, they don't need to be implemented in software at the operating system level. They can be used without root power and hook themselves into the keystroke stream data up at the application level. Anyone who's installed a keystroke macro recorder and playback tool, or an on-the-fly spellchecker - I have one running on my system - will realize that the application sitting in the system tray down there is monitoring everything they are typing. So it's not rocket science. It turns out also JavaScript running in our browser can monitor - and could alter, if it had some reason to, the text we're typing in, such as maybe delaying our own login - the flow of keystrokes

as we browse. That means that rogue JavaScript injected into a login page could recognize and steal usernames and passwords.

And let's recall that the integrity of our HTTPS connections is now being actively subverted by anything we allow to intercept and inspect our data, whether it's a corporate middlebox or locally installed AV that is wanting to help us by doing this. Those connections are no longer private. And of course banking trojans commonly include a keylogger module so they can try to capture our passwords when they recognize that we're logging into a bank that they have been set up to recognize and subvert.

And finally, as we know, keyloggers can be hardware. They can be a little device that you wouldn't even notice, a little lump in the cord connecting the physical keyboard to the back of the computer, a little what looks like an adapter at the back of the computer. In fact, I have some that are like, they are legitimate PS/2 to USB adapters. But I'm sure that isn't a keystroke logger. But such things could look just like a legitimate adapter. And we've actually also seen instances where the keystroke logger is molded into the USB connector at the end of a keyboard. So it's a nefarious little bit of malware or mal tech.

Second, data stealers. That's the term given to sort of a generic group of malware that gets into our machine and then, for their own purposes, goes hunting around our hard drives. Perhaps even around our whole network, if they're able to escape the confines of our machine. And those data stealers look for specific types of data that is worth something to the crooks that launched it.

Once upon a time, yes, those quaint vintage days of computing before the Internet, we mostly had true computer viruses which spread by themselves, often infecting floppies in order to get from one machine to another. They would typically, when they got into a machine, spew out emails containing an infected attachment in order to spread themselves. Back then, many viruses would search through qualifying files on our computers looking for text strings that matched a specific pattern, such as an email address. And then, by harvesting email addresses from everywhere they could find it, not only our own email client's address books, they were able to send things out to people that we'd never even contacted, but whose addresses appeared in documents, marketing material, saved on web pages or whatever. Those days are long gone. Times have changed.

But on the other hand, contemporary data stealers are much more sophisticated these days, and their interests are wide-ranging. Now they're searching out and looking for bank account details, ID numbers, passport data, credit cards and account passwords. They know how to recognize special files by names and by internal structure. Sometimes if files are weakly encrypted they're able even to decrypt them in order to get into them. So that would give them access to poorly constructed password vaults that contain our credentials and browser databases and may have authentication tokens and browser history. So several other classes of malware, such as bots and banking trojans that we'll be talking about in a second, also include sophisticated data stealing modules in order to get the data that they care about. So these data stealers are in the lexicon because they are a class of malware.

Then we have RAM scrapers because malware can't always find what it wants in the files on our computer, even if the malware has admin or root access. That's because some useful data, you might even argue some of THE most useful data, might only ever exist temporarily in RAM before it is deliberately scrubbed without ever being written to disk. Permanent storage of some data is now prohibited under regulation, such as the PCI-DSS, which is the Payment Card Industry's Data Security Standard, and Europe's GDPR, their General Data Protection Regulations. Those regulations prohibit the permanent storage of some information such as the CVV number used to authorize credit card

payments. And these regulations are naturally bad news for attackers because it means they can't easily get hold of those codes and the transactions that have already occurred.

However, with RAM scraping malware, which is able to keep an eye out for data as it is stored temporarily in memory, attackers may be able to spot what is critically useful data to them, such as CVV numbers and perhaps the accompanying full credit card information, and suck it right out of RAM, essentially scraping it. And as we know, computers must have, for a moment at least, a private key in RAM in workable decrypted state in order to perform decryption. So valuable secret data must transiently exist in RAM, even if only briefly. So things such as decryption keys, plaintext passwords, website authentication tokens, are being watched for by RAM scrapers. That's number three on our hit parade in the malware lexicon.

Then we have number four, bots. A bot, of course, is the term the industry has given to malware which sets up shop in a machine, then phones home to its bot master to report in. It requests and awaits for further instructions. And of course it's not alone. It's typically in a large network. It essentially establishes a semi-permanent backdoor through a computer so that the bad guys can later send commands from wherever they are.

And of course, as I said, we call a collection of such bots a "botnet." The other popular term for bot is "zombie" because they can also act like sleeper agents. The commands bots understand, including sending boatloads of spam from your IP address in order to avoid IP-based spam filters, searching for local files, sniffing for passwords, blasting other machines on the Internet, or rather blasting them off the Internet with floods of traffic, and even clicking on online ads to generate pay-per-click revenue span the range of what we have found bots doing. And of course they're also capable of receiving updates to themselves to allow them to be updated and to download new and improved modules.

So we've recently talked about the evolving strategies bots are using to sneak these updates into a network right under the noses of watchful AV screeners. And since bots are initiating connections outward to their bot masters, just like any IoT devices we've talked about where you just set them up and, oh, look, they're magically connected, these things work the same way. They don't need ports open for them. And our otherwise protective NAT routers don't protect us here because outbound connections by default are allowed, whereas NAT routers, as we know, drop inbound connections. So number four in the malware lexicon was bots.

Number five, kind of related, but much more of the specialist, are banking trojans. And they are more prevalent than this podcast would have you believe. I guess I don't cover them much because they're not that exciting. But they're definitely in existence. They deserve their own subclass of malware because of their specialization. They exclusively target their victims' online banking information.

And for what it's worth, I watch them grow in capability, where they're always adding more banks and lending institutions to their repertoire. As we might imagine, banking trojans typically include a keylogger component, which is used to sniff out the passwords as the user is using them to log into the bank. They 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.

And a trick widely used by banking trojans is web form injection. We haven't talked about this for years, but that occurs 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 given up any hope of actually understanding what's

going on when they use the computer. They just do what the computer tells them to. So they just hope for the best.

So when their bank suddenly appears to be asking for a credit card number for some reason, and/or their date of birth, they think, well, okay, the bank knows those things, so it's okay, I guess. Well, in the process they've just filled it into a form that the banking trojan added and populated on their login form, and it's collected information that it wanted that it wouldn't have otherwise been able to see them typing in because they weren't being asked. Anyway, banking trojans are nefarious, and they're also a real big deal as number five in the malware lexicon.

Second to last at number six is RATs. I wish it had a more distinctive acronym because it's never clear when I'm talking about a RAT, like, what I'm talking about. But it stands, of course, for Remote Access Trojan. It's got a lot in common with a bot, but it differs in that it's not usually part of a massive campaign simply to see how many bots can be corralled and managed for mass attack events. Instead, a RAT enables much more targeted and potentially much more damaging intrusion.

Normally a RAT is placed deliberately into a target machine, maybe through social engineering, a phishing attack, some other sort of intrusion. But here it's not simply intending to use the machine as a point of attack, but rather looking inward rather than outward, seeing what's there and essentially, being a Remote Access Trojan, giving the person who is operating it a persistent connection, able to sit there, look around, see what's going on.

They're able to take screenshots, listen to the audio in our rooms by turning on our computer's microphone, and then of course turning on the webcams, which always makes people nervous about whether or not it's possible for the cam to be on while the little webcam light is off. And we do know that there are software systems that allow the light to be turned off, even when the cam has continued to be on, which is why this podcast has long recommended using some sort of physical shutter, even if it's only a little corner of a Post-it note over the camera hole when you aren't deliberately sitting in front of it and having a conversation with someone.

And, finally, last but certainly not least, number seven in the malware lexicon is 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 what a bot or a RAT are, pretty much everyone at this point has heard the horrifying stories of entire municipalities, companies, or healthcare providers being shut down by ransomware and then having to pay a lot of money in order to regain access. So your typical user may not know exactly what it is, but they know it's a current problem, and it's one they don't want to have.

As we've observed last year, the enabling factor for ransomware, and actually this is just my own theory, I don't think I've read it anywhere else, I think the enabling factor for ransomware was the rise of readily exchangeable cryptocurrency because it used to be the way you caught the bad guys was you followed the money. Cryptocurrency makes that much, much more difficult. So until there was a safe and untraceable means for receiving payment, the ransomware business, such as it is, wasn't really able to take off.

But today there are even, as we know, companies specializing in being the go-between between a ransomware victim and the extortionists who are demanding payment. So this has solved the problem of I don't know how to send anyone bitcoins. Now there are helpful people that probably take a little piece of the action, but they know how to do that.

Ransomware attacks can be so lucrative that they are the inverse of the scattershot bot model. Attackers can afford to sit inside a victim's network, take their time to maximize the damage done. After finding their way into the network, they can set up and get ready to scramble hundreds or even thousands of computers at the same time. And if they discover that online backups exist, those have the opportunity to be compromised.

And what we've seen is, even if offline backups exist, if a large enterprise has done everything right, still the time and labor required to reimage and restore thousands of computers has been shown to be hugely damaging all by itself. No organization that is making responsible offline image backups expects to need to deploy all of them at once. They just figure, well, if a given machine suddenly explodes, we'll be able to get a replacement back up in no time. But much more difficult when ransomware takes all of the systems that it was able to find off and encrypts them at the same time.

And because the payout 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. It doesn't get a lot of attention, but the security forensics firms that are brought in often see effective defenses were in place and got sidestepped anyway. So it's just, you know, ransomware is the scourge of probably 2019 and promises no sign of letting up in 2020.

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 update, the last Patch Tuesday that Windows 7 and Windows Server 2008 R2 will be getting. So that will be something we I'm sure mention next week. Stay tuned.

JASON: Also bring in someone playing a sad violin song to go along with it, serenade out on that one. Good stuff. I love taking a look at some of these things that we hear the terminology, kind of don't think twice about it. It's nice to kind of get reintroduced to them.

Steve: Yeah. And it's weird, too, that, I mean, we really do have well-established classifications. There is some cross use, whereas like banking trojans are a thing, and they take advantage of keystroke loggers and data stealers, whereas those also exist just for their own sake. But, yeah, it's interesting. I mean, what we've seen is well-defined strategies for abusing our systems as a function of, ultimately, the way they're designed.

JASON: Sure, yeah. And what you said about cryptocurrency tied with ransomware, I completely agree with that. That really changed the game.

Steve: Yup. I remember once upon a time the way Russian bad guys would demand payment was you'd have to send them Western Union. You needed Western Union in order to wire the money to them. That was because that gave them some modicum of protection. But now, boy.

JASON: No.

Steve: With exchangeable cyber currency, unfortunately, that really did open up a new market.

JASON: Yup. Game has changed. Right on, Steve. Always good stuff. If anyone wants to go check out all that Steve is up to, you can do that, GRC.com. You'll find everything you need to find there. SpinRite, of course, Steve's hard drive recovery and maintenance tool. You can get a copy there. Also information about SQRL, growing information about SQRL to be found there. Audio and video of this show, as well as transcripts, which can only be found there. Of course we host audio and video of this show on our site. But if you want transcripts of the show, go to GRC.com, and you can find that there.

Our site is TWiT.tv/sn. That's the show page for Security Now!, the official show page here at TWiT. There you can find everything that you need to know about this show minus the transcripts. Audio and video, play it in the page or subscribe. All the links are there for you to subscribe in a number of different ways, a number of different places. And you'll also find there the kind of posting days, every Tuesday.

We record live every Tuesday at 1:30 p.m. Pacific, 4:30 p.m. Eastern, 21:30 UTC. But if you're subscribed, it kind of doesn't matter. You don't need to catch it live. Most people get it in their feeds automatically. That's the beauty of podcasts. So go to TWiT.tv/sn and subscribe, if you haven't already. And then you don't even have to think about it. Steve appears like magic, as he should, into your ears.

Steve, thank you so much for letting me bomb into your show once again. I really appreciate it.

Steve: Jason, glad to have you while Leo is in Vegas. And we'll see you next vacation, I'm sure.

JASON: That's right. We'll see you next time on Security Now!. Bye until next week.

Steve: Bye.