Security Now! #886 - 08-30-22

Wacky Data Exfiltration

This week on Security Now!

This week we begin by discussing the implications of last week's LastPass breach disclosure. We then look at some recent saber rattling by the U.S.'s FTC and FCC over the disclosure of presumably private location data. I want to share some pieces of a fascinating conversation with a Russian ransomware operator, gaining some insight into the way he conducts attacks and the way he views the world, and I want to tell everyone about a new tracking-stripping and privacy-enforcing eMail forwarding service that's just come out of a year-long beta from the DuckDuckGo people. We also have another big and widespread IoT update mess to share. Then I have some welcome progress to report about my work on SpinRite and some listener feedback. Then we're going to look at some recent goings on at the Ben-Gurion University of the Negev which never fails to entertain.

https://www.goarmy.com/careers-and-jobs/specialty-careers/army-cyber.html
https://twitter.com/ArmyChiefCyber/status/1563295564380250113

Security News

LastPass Breached

The news of last week's LastPass breach announcement overwhelmed my Twitter DM's, so I'm leading with that news since so many of our listeners, myself included, are still using LastPass. I had, therefore, received an eMail from LastPass too. The current LastPass CEO, Karim Toubba had this to say in their online blog posting which echoed the eMail they sent to everyone:

> *I want to inform you of a development that we feel is important for us to share with our LastPass business and consumer community.*
>
> *Two weeks ago, we detected some unusual activity within portions of the LastPass development environment. After initiating an immediate investigation, we have seen no evidence that this incident involved any access to customer data or encrypted password vaults.*
>
> *We have determined that an unauthorized party gained access to portions of the LastPass development environment through a single compromised developer account and took portions of source code and some proprietary LastPass technical information.*
>
> *In response to the incident, we have deployed containment and mitigation measures, and engaged a leading cybersecurity and forensics firm. While our investigation is ongoing, we have achieved a state of containment, implemented additional enhanced security measures, and see no further evidence of unauthorized activity.*
>
> *Based on what we have learned and implemented, we are evaluating further mitigation techniques to strengthen our environment. We have included a brief F.A.Q. below of what we anticipate will be the most pressing initial questions and concerns from you. We will continue to update you with the transparency you deserve.*
>
> *Thank you for your patience, understanding and support.*

The short version of the five F.A.Q. questions and answers which followed was that there is zero impact upon LastPass users — no need to change passwords, do anything, or take action of any kind. I'm sure they are unhappy that this occurred since I'm sure that they hold their proprietary information in high regard and don't want attackers snooping around in it. But we've always known, since I first checked out the technology that Joe Siegrist originally designed, is that so long as the LastPass code that runs our local browser vault is not **itself** compromised, the only thing we're providing to LastPass, the only thing they have of ours to lose, is a very well-protected encrypted blob of entropy. That's what they hold for us in the cloud which allows them to link all of our devices together. And I'm sure this is not unique technology. Though I haven't looked, I would imagine — and hope — that's what every other password manager also does, because it's the only way to do what we all want, safely.

LastPass uses a strong, many-iteration PBKDF — password based key derivation function — which runs in our local browser to encrypt all of our password data before it ever leaves our local machine. So, you need to have a good strong password to protect your vault. If you have that you're as safe as you could be. And presumably, adding any of their other security measures

such as multi-factor authentication, hardware dongles, etc. only strengthens things from there.

But this leaves us with the question: With LastPass having admitted to having one of their developer accounts breached, should we change password managers? That's certainly a worthwhile question. Lacking any additional information — and no additional information is available — I think that's an emotional decision rather than a rational decision. The reason I think that, is because there's no factual basis for knowing about what matters. To make an informed decision it would be necessary to deeply understand the company's policies and procedures and to know exactly how this particular breach occurred. Their policies and procedures would tell us how they have set up the barriers, which hopefully exist, between their developer resources and their production services. But then you would also need to know that same thing about the password manager you were considering switching to. Again, an emotional decision needs no justification whereas a rational decision is only about justification.

I've always been careful to draw a clear distinction between policies and mistakes. Policies are deliberate; mistakes — well they're mistakes. When you're an employer and an employee screws up, do you fire them because they screwed up? Or do you consider that they made a mistake and have learned a valuable lesson? If, as a consequence of having made a mistake they're now a better and more valuable employee, why give them to your competition?

Unfortunately, we don't know enough about the inner workings of LastPass to make an informed decision about switching. Should we now be more or less afraid? How does their actual policy and behavioral security **after** this incident compare to the actual security available elsewhere? If LastPass learned a valuable lesson, that's great. But I have no idea... and neither does anyone else. Their track record is all we really have to go on. And it's been good so far because the security architecture is sound, and that's what I'm relying upon. At the same time, presumably everyone else's security architecture is equally sound, because none of this should be rocket science anymore.

If I were starting out today, all other things being equal, I would probably choose BitWarden. Being open source I'd be able to do the same sort of security architecture vetting that I once did with LastPass' designer, Joe Siegrist. As we all know, BitWarden is currently a sponsor of the TWiT network, and I think that's great, though it's worth noting that LastPass had never been a sponsor here at the time I chose them. I chose them simply because Joe was more open than everyone else, which allowed me to understand exactly how their system worked and why it was the proper design.

Many of the flood of DM's I received last Thursday asked whether I was still using LastPass, and if so, whether I was now planning to change. Security Now! Podcast #256, dated July 9th, 2010 was titled "LastPass Security." The little summary description for it on TWiT says: "Steve thoroughly evaluates LastPass, explains why high-security passwords are necessary, and tells us how LastPass makes storing those passwords secure."

So it looks like I've been using LastPass for the past 12 years, and I still am. If they ever give me a rational reason to change, I will—in a heartbeat—and whether or not BitWarden is a sponsor of the TWiT network at the time, I would probably go there because openness matters. But so does inertia and the devil you know.

The US Federal Trade Commission (FTC)
Yesterday, the US Federal Trade Commission filed a lawsuit against the large data broker Kochava. The lawsuit's complaint alleges that the company offered for sale the precise geolocation data of hundreds of millions of mobile devices — revealing potentially sensitive information in what the agency says amounted to an unfair or deceptive consumer practice.

According to the FTC complaint, as part of its operations the Idaho-based Kochava "collects a wealth of information" about people and their mobile devices, including by purchasing it from other data brokers, and sells customized feeds.

The FTC explained that among the information it sells is precise geolocation information associated with a unique marketing ID that can be used to reveal visits to sensitive locations, such as places of worship and healthcare providers. Such data can also be relatively easily tied back to an individual by observing patterns, such as regular sleep and work locations.

Samuel Levine, Director of the FTC's Bureau of Consumer Protection said in a press release announcing the suit: "Where consumers seek out health care, receive counseling, or celebrate their faith is private information that shouldn't be sold to the highest bidder. The FTC is taking Kochava to court to protect people's privacy and halt the sale of their sensitive geolocation information."

When asked about the suit, Kochava's legal representatives did not immediately respond to a request for comment. Kochava charges clients $25,000 for access to its location feed, and until recently offered free samples.

Kochava attempted to preempt the action by suing the FTC earlier this month, alleging overreach in proposed complaints the agency shared in July and August. Shortly before suing the FTC, the company also announced a new capability called "privacy block" which it said should assuage the agency's concerns by removing "health services location data from the Kochava Collective marketplace."

This is all being allowed to occur only because it's invisible to the consumer. If tracking was apparent it would never have grown so out of control. As we know, Apple started requiring apps to obtain consumers' explicit permission to track them, and the result was a resounding "No!" I'm glad that slimy companies like this Kochava are finally being put under the spotlight. It's annoying that it took the Supreme Court's overturning of their previous decision in Roe to bring this to the forefront, but better late than never.

NOTE: When I attempted to go to https://www.kochava.com/ to see what they were bragging about, Chrome told me that the domain was unknown: DNS_PROBE_FINISHED_NXDOMAIN. Then I smiled when I realized that was because I took Leo's suggestion last week and decided to experiment with NextDNS as an advertising and tracking blocker. Obviously, they already know about Kochava... and they're saying "ahhhhhhh... No!"

The Federal Communications Commission (FCC)
Not to be left out, thought not that it appears to matter much, the US's Federal Communications Commission has launched an investigation into mobile carriers' geolocation data practices.

Last Thursday, the FCC shared responses from mobile carriers to a probe into how they handle geolocation data, and announced a new investigation into carrier compliance with the Commission's rules about disclosing how such data is stored and shared.

The FCC's Chairwoman Jessica Rosenworcel said in a press release: *"Our mobile phones know a lot about us. That means carriers know who we are, who we call, and where we are at any given moment. That's why the FCC is taking steps to ensure this data is protected."* Uh huh. Good luck with that. Though I suppose this might answer the question of where slimeball Kochava obtained the information they're aggregating and reselling.

The Commission sent inquiries to 15 carriers, including AT&T, T-Mobile, Verizon, Google-Fi and others last month asking them to spell out their policies around geolocation data, including how long information was retained, as well as how and why it might be shared with third parties. The FCC requires mobile companies to get consumer consent for sharing information, unless such sharing is necessary to complete a service or required by law. Unfortunately, has anyone ever actually read the fine print of the agreement with these companies? And what is one to do if the terms are onerous? All other carriers use the same fine print.

Two years ago, in 2020, the FCC proposed more than $200 million in still-pending fines for major carriers for selling user location data without consent or appropriate safeguards. Jessica Rosenworcel tasked the FCC's Enforcement Bureau with the new investigation into the companies' compliance with rules requiring them "to fully disclose to consumers how they are using and sharing geolocation data." Again... a lot of good that will do.

Justin Brookman, the head of tech policy at Consumer Reports said that, nevertheless *"The quality and specificity of answers definitely ranges among the respondents, but there's some interesting, concrete information in there, especially on data retention periods."* However, not surprisingly, in some cases the responses simply referred to their dense, publicly available privacy policies. But some others did answer questions directly point by point.

But the Consumer Reports guy agrees with me that transparency isn't enough. He said: *"People have no choice but to share very sensitive data like geolocation with mobile carriers for those products to work. There should be substantive constraints on what they do with that information and for how long they keep it."* In a world where that information can be sold to third parties in real time, and where its timeliness makes it valuable, it's unclear to me whether "retention time" matters at all.

Harold Feld, the Vice President of Public Knowledge also called for regulatory action, saying the FCC should "set new rules of the road" for mobile subscribers' privacy. He said: *"These letters show that, despite the constant invocation of 'industry standards' and 'best practices,' carrier geolocation data practices are all over the map."* For example, the length of time carriers retained location data — as determined by proximity to cell towers — ranged widely, and was as long as five years in the case of AT&T.

Unless we make them delete it they're not going to. But, again, if they are allowed to sell it immediately to third parties, retention no longer matters at all. They claim that they must keep it for business purposes and to maintain the health of their networks. Fine. Simply outlaw its sales to any other entity. But the FCC appears to be toothless to me. They completely ceded control over broadband privacy during the Trump administration. And while the FCC still theoretically has substantial regulatory authority over mobile phone carriers, the carriers appear to simply be ignoring the FCC.

California, here I come!

And in very late breaking news, Techdirt's Mike Masnick reported a couple of hours ago that the California State Senate had just passed what he described as three horrific new Internet regulation laws, apparently written by bureaucrats who have no idea how the Internet works. Since this just happened, I haven't had the chance to look into it. But if it's interesting, I will for next week.

A conversation with a Ransomware Attacker

The guys over at the publication The Record had a lengthy conversation with a Russian Ransomware attacker by the name of "Mikhail Matveev." Although I didn't think that much of the conversation, which revolved around the squabblings among adversarial ransomware groups, would be of much interest to our listeners, Mikhail's answers to a couple of the questions were interesting. So I've selected a few bits to share. I should mention that this is a translation from Russian, so the semantics will be a bit non-English. And I've also edited it since this young man's choice of descriptive language can be a bit blue.

Dimitry from The Record: How often do people from different affiliate programs compete in the same network to extort victims? Have you had such situations?

Mikhail: This happens often. Especially when several people own the exploit, or pour logs from the same traffic market if we are talking about extracting initial access credentials with a stealer. I took some source codes, so-called proof-of-concept, from GitHub and modified them. If you remember, there was a well-known CVE for the Fortinet VPN. We found it with one programmer from the forum. Based on the list of IP addresses, we got approximately 48,000 entry points. I was very surprised then, really shocked. But we did not even work through 3% of this list. Not enough time.

And when others — well, let's say our competitors — began to use this vulnerability, there were intersections across networks. I often went into a network already locked by someone and didn't touch them, because it's not my job to encrypt for the second time, but some guys overlocked networks. They come in and see that it is encrypted and so that nobody gets it they encrypt it again. There were cases when the guys and I just crossed paths on the network during development, exchanged contacts, and somehow discussed what to do next. We basically always agreed.

And it even happened that we then jointly did some other projects. In the summer of 2022, this happens all the time, because everyone is hungry for the material. How can we get to the initial access? Actually, there aren't many options. There are vulnerabilities, such as RCE in various products of VPN devices, everything that can give access to the network. Or a network access login from stealers. But basically, everyone is now flooded from traffic exchanges and there is little unique traffic. And those who have it, they pour just for themselves or are already working in some teams, so it's absolutely normal that there is a conflict of interest on the networks and now it will be even more.

So, I thought Mikhail's comment that they were only able to exploit 3% of the list of 48,000 Fortinet VPN's because there's not enough time. In other words, there really is an active race when a new patch drops for a critical remote access vulnerability. These cretins are actively watching everything waiting for the first glimmer of a newly discovered problem.

And, significantly, they are **not** finding any of these problems themselves. These are not high end security researchers gone bad. They are living off of the interval in the delay to patch. They're not good enough to find the trouble themselves. But they are good enough to quickly weaponize a working proof of concept then immediately turn around and employ it to gain entry.

Dimitry: Tell me about some attacks that stood out to you. Which was the fastest? How long did it take from the first penetration into the network to receiving the payment?

Mikhail: There were many interesting ones. But I would like to sum it up, before talking about the attacks. There are small networks, there are medium networks, and there are very large networks. And I'll tell you, it's much easier to work with a network of an organization with $1 billion revenue than in a network of an organization that has an income of $9 million. I'll tell you why. There are many more computers that are easier to hide on and easier to navigate than in a small network where you are limited. You have to move very fast. And when I started my career, I started with BlueKeep — a vulnerability in Microsoft Remote Desktop. I hacked five small networks per day because I had to go in and do it right away. But, as I progressed, the time I spent on the hacks increased.

My longest development… Probably everyone has heard about the Capcom company. I got there through a Fortinet vulnerability. As a matter of fact, when I went there, I was a little surprised that everything was in Japanese. There is no hierarchy, there is no division into departments, and they have everything in a heap. I found a dead domain admin. That is how the name Babuk appeared. Capcom had an admin Babak, or bambook. And when I found this administrator, I realized that no one uses him, but he was an enterprise type.

So, Mikhail is explaining that he found an abandoned active administrative account which he was able to use. And he took "Babuk" as one of his several aliases.

The fastest attack in my life happened as soon as I got the ProxyLogon vulnerability. At that time, I had a programmer on a grant who was finalizing the exploit. One of the interesting networks was a logistics company in the Netherlands. Large warehouse. Very large warehouse.

I got in and immediately obtained the domain admin tokens. These guys weren't very security conscious and didn't worry about anything. I remember I went there at 8:00 pm Moscow time and at about 4:00 am Moscow time it was already all locked up. From 6 a.m., the administrator wrote to us in a panic, to which I told him, bro, wait for the supervisor.

Looking around the network, everything seems to be simple and clear. They have an administrator's domain for us. The password was the same for everything, on hypervisors, on a backup server in the work group. After analyzing the network, I found a WIM backup system. I could get all the passwords from it and thereby got all the backups, although their backups were so bad. They just backed up to the NAS. I went to the NAS and formatted it. Went to ESXi, encrypted, and then after about an hour, he wrote to us.

The admin wrote right at midnight, "I would like to resolve the issue." I said that the issue could not be resolved, because he was not a boss. In the morning I had to fly to another city. I remember sitting at the airport, the company writes to me: $2 million. Transferring $2 million. I have never had such an amount in my wallet. I get on the plane, realizing that I have a laptop with $2 million. Well, I gave them decryptors, and when I arrived, I opened the chat.

Damn it, something is not right there, they just yell: You destroyed VMDK. I tried to figure it out and asked for VMDK samples. But the VMDK files are zero KB. So everything is screwed. I am writing to this developer who created the exploit for me. "How could this happen?" He says, "Well, I don't know", he said, something broke. And they asked to return the money. Well, we had no choice but to block them. We scammed them for this money. I still blame myself for this. It was the fastest and most solvent attack I've ever done.

Dimitry: How do you see the ransomware industry in three years? Will ransomware remain the best monetization model for cybercriminals, or will they move to something else?

Mikhail: It's like how carding used to be popular and there was a lot of money in it, but now it's dead. And ransomware will soon die — not in three years, but sooner. Literally, everything has changed over the last six months. ***Since the beginning of the special operation in Ukraine, almost everyone has refused to pay.*** *I often encountered people who wrote to me in the chat, "You are a Russian occupier. Be content with $10k. And we won't give you more. At least take that." Return on investment has completely fallen in the last six months. It became difficult to work, in general.*

If it dies, it dies. You need to come up with something new. But ransomware is worse than heroin. I haven't tried heroin, but I've seen people who are on it, and I'll tell you this: ransomware is worse than drug addiction. There is no such money anywhere as there is in ransomware. I even compared it to drug dealers from hydra, the world's largest dark net marketplace, which was shut down this year. They earn less than we do.

But at the moment, ransomware remains the leader in monetization. There are no other schemes on the internet that would carry more monetization. Or I don't know about them yet.

As I read that I'm struck by how casual Mikhail is about being a criminal. There's an utter lack of morality. He did appear to feel badly that his decryptor didn't reverse the encryption of the large warehouse's VMDK files. So he got $2 million without returning their data. But what seems to be utterly absent is the idea that extortion itself is wrong. He talks of this using the term "career." Like it's a legitimate profession. Like his parents would be proud. As though, if you have leverage over someone, using that leverage for your own personal gain, at their loss, is acceptable.

Anyway, I thought that everyone would find some of this interesting. These guys are not geniuses. They're computer savvy, using other people's tools to force those abroad — as in not in Russia — to give them money.

DuckDuckGo's Privacy-Enhanced eMail Forwarding

The privacy centric DuckDuckGo has had what it calls an "Email Protection" service in beta since July of last year, 2021. But they have just opened it to the public. It looks like a very useful and completely FREE service. So our listeners might want to jump over and grab their name or their favorite handle quickly before it's taken. I'll explain how to do that, then I'll tell you why this seems like a nifty service...

To register, go to https://duckduckgo.com/email/. If you don't currently have their browser extension installed you'll need to do that first. You can remove it later since it's not required to use the service. You'll be asked to provide a username which has not yet been taken. It will be your eMail @duck.com. You also provide an eMail address which will receive cleaned and forwarded eMail.

Okay, so what does this all get you?

Their Email Protection is DuckDuckGo's dedicated email forwarding system which strips advertising and profiling trackers from emails — links, scripts, images, media, etc. — before forwarding them to your registered forwarding eMail. When you receive the forwarded eMail you'll also see a short report of how many trackers were removed, which companies were responsible for their injection, and more. DuckDuckGo says that running the beta program for a year revealed that over 85% of all emails on the tester's communications contained trackers of one form or another.

So anything sent to "username@duck.com" will be forwarded after being cleaned and reformatted. And, in a VERY cool feature, Email Protection also provides users with unlimited disposable and manageable private addresses to use on sites you want to supply with a per-site eMail address. These can later be deactivated if spamming to that address becomes annoying. You can ask for as many throwaway eMail addresses as you need. And, of course, using unique eMail addresses confers some of the benefits of using unique passwords: In the event of a website's data breach linkability of your identity will be dramatically reduced.

Messages passing through DuckDuckGo are never stored by the vendor — they make that very clear — while what small amount of account and forwarding information is kept for operational reasons is deleted within 30 days after the account's closure.

And even though eMail is forward to your real eMail address, it's possible to reply to those eMails from a Duck address. This can be useful where anonymity is useful.

The FREE service has a user-friendly dashboard for quickly configuring forwarding addresses, making on-the-fly changes, managing account settings, and more. In addition to a browser extension there are apps for Android and iOS which allow for the same sort of dashboard management.

I still don't like the name, but the service seems pretty cool. Since I run my own server I'm able to create eMail aliases without end. But that's not generally available and this service does that and a lot more.

Another IoT Mess

So, we have another big IoT mess. Each one of these is a lesson. I'm going to try not to get preachy this time since I know it can get tiresome. But the state of the IoT industry brings my blood close to a boil every time. So, here's what's going on this time:

The cyber security firm “CYFIRMA” recently published a report describing a long and still outstanding security threat created by insecure Internet surveillance cameras produced by a US-based firm “Hikvision” located in Southern California. One year ago, in September of 2021, in response to a discovery by security researchers which was given CVE-2021-36260, Hikvision did the responsible thing: They published a firmware update to correct a serious vulnerability.

The researchers discovered that the Hikvision cameras were vulnerable to a critical command injection flaw that's easily exploitable via specially crafted messages sent to the camera's vulnerable web server.

That was then. Today, CYFIRMA analyzed a sample of 285,000 Internet-facing Hikvision camera web servers. They found that, today, roughly 80,000 are still vulnerable to exploitation. And this, of course, is the problem. Some contractor you've hired purchases a camera, or 50, and installs them. They set them up, send you an invoice and move on to their next job. Meanwhile, you have some number of web servers on your network which can be taken over remotely. And that take over is not theoretical.

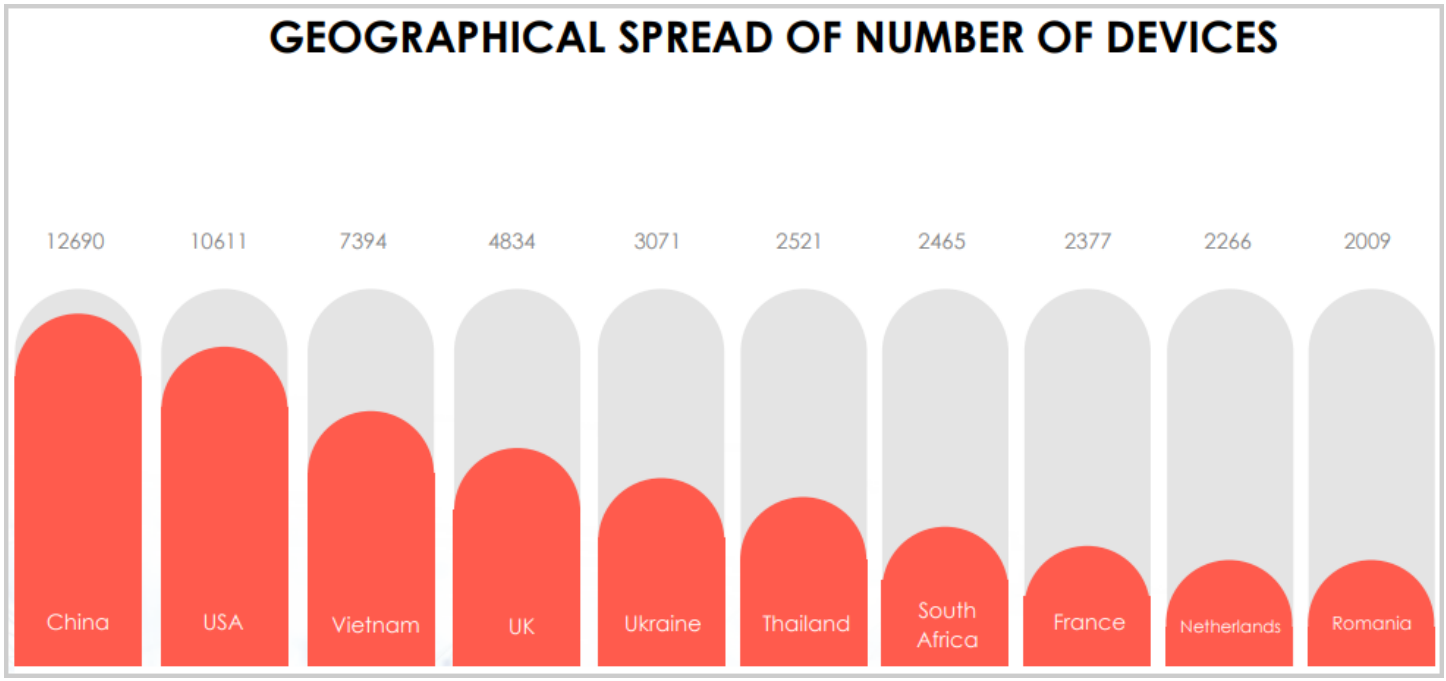
There have been two known public exploits for CVE-2021-36260 for that CVE, one published in October 2021 and the second in February 2022, so this is exactly the sort of thing that our friend Mikhail in Russia surfs the Internet looking for. In December 2021, a Mirai-based botnet called 'Moobot' used one of those two exploits to spread aggressively and enlist those systems into DDoS swarms. What's generating those record-breaking DDoS attacks which now force everyone who needs to remain online to move behind and to pay for protection? It may include thousands of compromised Hikvision cameras.

In January of this year, CISA alerted that CVE-2021-36260 was among the actively exploited bugs in its list. CISA warned organizations that attackers could "take control" of devices and that they should be patched immediately. How'd that work out? As I noted, those 80,000 still-vulnerable surveillance cameras were just recently enumerated. The cameras are very

popular and they appear to be industrial grade. Hikvision has an impressive website. In CYFIRMA's report they tracked those 80,000 vulnerable IPs back to 2300 organizations across 100 countries. None have applied the security update which is nearly a year old.

CYFIRMA's report notes that Russian-speaking hacking forums often sell network entrance points relying on exploitable Hikvision cameras that can be used either for "botnetting" or lateral movement to gain entrance into the organizations where they are deployed. Like I said, just what Mikhail is looking for.

I have a chart in the show notes showing the geographic breakdown of the camera's locations. Most of them are located in China (with 12,690) and the United States (with 10,611), while Vietnam, the UK, Ukraine, Thailand, South Africa, France, the Netherlands, and Romania all count between 7,000 and 2,000 vulnerable endpoints each. There's a bunch of them.



The exploitation of the flaw doesn't follow a specific pattern because many threat actors are vying for access to these camera resources. And this doesn't even touch on what all of those camera might be seeing. Who knows what they are protecting and also revealing?

And, as if a largely unpatched remote command injection web server vulnerability wasn't trouble enough, there's also the fact that the device's ship with weak default passwords that are often never changed after the cameras are set up. Hacking forums often offer lists of credentials for Hikvision cameras.

I still doubt that the public at large understands the danger that's represented by the casual attachment of high-tech devices to their network. And as long as that remains true, purchasers won't know that they need to consider the operational life cycle of such devices. Everyone has been trained now about OS and Smartphone updates. This needs to be broadened to somehow include everything else.

SpinRite

Over this past weekend I posted two status updates to the grc.spinrite.dev newsgroup:

The first had the subject: Friday Night Update

> *Gang,*
>
> *I just finished the complete read-through of SpinRite's DynaStat system. I've been slogging my way through it for the past week or so. It's extremely involved and it was working once. I wanted to be certain that I hadn't done anything to break it with all of the changes I've made — the I/O driver abstraction and the relocation of several working buffers into high memory. Since they did affect the DynaStat code deeply, I've had to work my way through every code path. It's still going to need extensive testing but that will be joyful since it will mean that SpinRite is essentially working and just needs to have the final bits of debris eliminated.*
>
> *With this done, I now need to finish the comparatively trivial task of updating the rest of SpinRite's main processing loop, the data inversion media testing, etc. ...and then it will be ready for the thorough testing of all of its main data recovery loop.*
>
> *But... we're definitely getting tantalizingly close. :)*

And then: Saturday Night Update

> *Okay... I'm done.*
>
> *This is =NOT= to say that I have =ANY= illusions that it could POSSIBLY run yet. There's no possibility. But I have finished working through all of the code and now it'll be up to SpinRite to show me where it's not yet ready for prime time.*
>
> *What I plan to do next is to get it actually running so that it would APPEAR to the casual observer to be working. That'll still be a chunk of work since I've deliberately not allowed it to begin execution. It's certain to explode fabulously. But before long, it won't be exploding anymore when it runs.*
>
> *At that point, when there's no longer anything obviously wrong, I'll verify that it's actually doing something useful and that all of the various data recovery paths — several of them new — are working as they are designed to.*
>
> *And **THEN** it'll be done.*

So I just wanted to share with everyone here, who is **not** following along with the blow-by-blow in the SpinRite development group, where things stand. This evening I will begin running SpinRite and fixing everything that doesn't run. Once everything appears to be running, I'll then begin the work of carefully inducing various sorts of media read and write failures and carefully watch SpinRite deal with each type problem.

A listener of ours, Ameel Khan (@ameel) sent me a Twitter DM:

> *Hi Steve! Love the show; been a regular listener for 16 years now. Check out this video of John Carmack talking about the importance of using a debugger while you code.*

The YouTube video Ameel linked to is an interesting 15-minute conversation with the legendary coder John Carmack. I have the link in the show notes and it's our GRC shortcut of the week, so https://grc.sc/886 (https://youtu.be/tzr7hRXcwkw).

What I learned by watching the video is that John and I code in exactly the same way. At the beginning of my return to working on SpinRite, everyone here heard me talking about setting up a comfortable and smooth debugging environment before I did anything else. And you've heard me mention it over and over since then. My wife Lorrie, lived with me grumbling about that for several months while I struggled to get everything working exactly the way I wanted. In my case it was challenging because my target environment was MS-DOS and to do the sort of debugging I wanted to do I needed a real-time link between a state of the art 64-bit Windows machine and a 16-bit real mode DOS machine. That has become much more tricky as the years have separated these two worlds.

Anyway, I thought it was interesting that John's code writing philosophy and mine are identical. Rather than trying to guess what's going on, rather than attempting to debug in our heads, we both immediately go to the debugger to watch the code execute step by step. As I've often noted here, something about the programmer's ego prevents us from seeing what the code actually does. We see what we want it and expect it to do... right up until the debugger slams our face into the reality. At one point John notes that tools that are easy to use get used, whereas tools that are difficult or cumbersome tend to only be used as a last resort. He and I have apparently both learned the lesson that having a comfortable and easy to use debugging environment is the way to get the best possible code written.

So, thank you, Ameel for sharing the link to that interesting conversation.

Closing The Loop

Vlad Jirasek / @VJirasek

> Hi Steve, I have an update on this. I pressed Cybereason to clarify whether the escalation of privileges would have been successful if the users were NOT part of the local Administrator group, and they confirmed it. See: linkedin.com/posts/cybereas…
>
> Might be good to mention on next SN. Even Microsoft is saying that removing admin privileges makes over 90% of attacks in effective. Thank you

Q: Very nice report, thank you. However, may I ask why you do not mention recommendation for computer users not to be assigned Adminisyrartor privileges as one of the key controls protecting them against the eslacation of the attack? If an user is not member of local Administratro group then running fodheper.exe will not give attackers the administrative privileges by bypassing UAC. Am I correct?

Cybereason: "Vlad Jirasek Thank you, we should have previously addressed that the point of the article is not to be exhaustive in terms of recommendations. In the case involving Bumblebee, users were already in administrator group and UAC bypass worked but you are correct, users need to be in administrator group. The article is focusing on post exploitation, the recommendations list is not exhaustive."

Ed McKiver / @OhWellDamn2010

Hi Steve, FYI, I canceled my LastPass Premium subscription today (due to the recent close-call security breach). I've had LastPass since they were a sponsor on TWIT, you gave your thumbs-up to their software/encryption, and before LogMeIn purchased the company. I'm trying to limit my exposure with my password managers now to just one. I've used Passwords Plus from DataViz since ver. 1.0 when it was sold on 5 inch floppy disk. They recently stopped all support for their product and their CEO decided to not move over to a subscription option to keep it profitable. I tried mSecure Premium as the recommended password manager to replace PasswordsPlus, but decided to cancel that Password Manager today too, as I found their tech support severely lacking. It seemed to me that mSecure was a one or two man operation. I'm sticking with BitWarden as they have the best options, prices, and also support the Yubikey.

Thank you as always for your great work on Security Now! I'm looking forward to SpinRite 6.1 since I've been a subscriber since ver. 1.0. / Ed / Redlands, CA

Wacky Data Exfiltration

Through the years, we've had fun considering all of the various ways that Dr. Mordechai Guri and his student researchers at Israel's Cyber-Security Research Center of the Ben-Gurion University of the Negev, have come up with for secreting information from Air-Gapped computer equipment.

We'll all recall picking up the vibrations from the surface of a bag of potato chips sitting unnoticed in a conference room... and there have been many such inventions, all of which they have developed and actually performed in order to determine the feasibility and achievable information transmission rate. So in the past week we have their reports of two additional covert information leakage channels.

The first is actually one that we have discussed in the past: The blinking LEDs on network interface cards. I was quick to discount it since the LEDs don't actually blink in time with the data. Although there is no standard for their operation, these devices arrange to show a more or less blinking light when there is activity on the line, and in either direction. But that didn't deter the intrepid Israeli researchers.

In their paper titled: *"ETHERLED: Sending Covert Morse Signals from Air-Gapped Devices via Network Card (NIC) LEDs"* they explain: *"Highly secure devices are often isolated from the Internet or other public networks due to the confidential information they process. This level of isolation is referred to as an 'air-gap .' In this paper, we present a new technique named ETHERLED, allowing attackers to leak data from air-gapped networked devices such as PCs, printers, network cameras, embedded controllers, and servers. Networked devices have an integrated network interface controller (NIC) that includes status and activity indicator LEDs. We show that malware installed on the device can control the status LEDs by blinking and alternating colors, using documented methods or undocumented firmware commands. Information can be encoded via simple encoding such as Morse code and modulated over these optical signals. An attacker can intercept and decode these signals from tens to hundreds of meters away. We show an evaluation and discuss defensive and preventive countermeasures for this exfiltration attack."*

Okay. So they're cheating. Or at least they're modifying the rules in a kobayashi maru-like way. They're allowing for malware to rewrite the NIC's firmware to take control over the LEDs. In that case it would, indeed, be possible to hugely increase the rate at which data could be exfiltrated from an air-gapped network which has no other means of communicating but whose NIC cards can be seen.

What I appreciate most about these guys is that in every case they really do wrestle to ground whatever wacky topic and method they are researching. They really do the work. For example, in this case, their eight page paper described the three methods which can be employed to control the LEDs of NIC interfaces:

1) Driver/firmware control: In this method, the LED-controlling code runs as a kernel driver or within the NIC firmware. Changing the LED state/color requires direct access to low-level registers or special non-volatile memory addresses. This method enables the highest degree of control over the LEDs but is very hardware-specific and mostly undocumented. For example, documentation discusses how to control the Ethernet LEDs in an Intel NUC PC. It can be done from a kernel driver or by writing to specific addresses in flash memory (word 0x18), which holds the LED's configuration. For embedded controllers, the control on the NIC is typically performed via internal BUS or USB interfaces. For example, sample code for LAN915X Ethernet controllers programs the corresponding LED register (LED GPIO CFG) via USB commands.

2) Link status control: In this method, only the status LED can be controlled. The malicious code can intentionally change the link speed, which in turn causes the network adapter to change the status LED. For example, setting the link speed to 10Mb, 100Mb and 1Gb will set the status LED to off, green, and amber, respectively. Selecting the link speed can be done by interacting with the NIC driver. For example, the ethtool command-line tool in Linux enables to change of the link speed of the Ethernet controller. The same is possible in the Windows OS via the netsh command. Note that setting the link speed requires root/admin privileges in both the Linux and Windows OSs. Technically, the link speed is determined through the autonegotiation procedure. In this procedure, which occurs in the physical layer, the connected devices share their capabilities regarding supported parameters such as transmission rate, half/full duplex, etc. The link speed of the network card (NIC) can be determined from the computer's OS.

3) User LEDs control: In this method, the user directly turns the status LEDs on and off by enabling and disabling the Ethernet interface using API or tools such as the ethtool or eth. The user directly turns the status LEDs on and off by enabling and disabling the Ethernet interface using API or tools such as the ethtool or eth. Another technique to blink the status LED is using the 'test' or 'identify' functionality, enabling the operator to identify the adapter by visual indication. These operations can be triggered programmatically or via low-level tools such as ethtool.

In wringing out every detail, they actually implement all of this. They consider the cameras needed to receive the information, and what can be done to make that end work as well as possible. Then they finally get down to the effective bitrates which are achievable through each of these three methods:

Technique	Modulations /	Bit rate	Relevant information
Driver/firmware control	OOK, Blink-frequency, colors	100 bit/sec	Text files, Username/passwords, encryption keys (e.g, RSA 4096), PIN codes
Link status control	OOK, Blink-frequency	1 bit/sec	Username/passwords, credentials, encryption keys (e.g, AES 256), PIN codes
User LEDs control	OOK, Blink-frequency, colors	2 bit/sec	Keylogging, username/passwords, credentials, encryption keys (e.g, AES 256)

(OOK = On/Off Keying)

Okay, so that's the first of the two wacky ways of exfiltrating data from an air-gapped machine. If you cheat with firmware then, yes, you can squeeze out 100 bits per second and you could do some damage. But it's worth remembering that some of the best-kept secrets are also very short. A server's elliptic curve private key might only be 256 bits long. And even a larger RSA key is still only 2 or 4K. So even at a measly 1 bit per second, sluggishly bringing a LAN link up and down, a 2048-bit key can be transmitted in only a little over half an hour, about 34 minutes.

The second wacky idea might be somewhat less wacky. Their recently published 11-page paper is titled: *"GAIROSCOPE: Injecting Data from Air-Gapped Computers to Nearby Gyroscopes"*

They spelled "gyroscope" GAIROSCOPE to get the word "AIR" in there, since it's an air-gap attack. Now you might think "Gyroscopes? What!?!?!?" but they're in every one of our Smartphones. The paper's abstract explains:

> *It is known that malware can leak data from isolated, air-gapped computers to nearby smartphones using ultrasonic waves. However, this covert channel requires access to the smartphone's microphone, which is highly protected in Android OS and iOS, and might be non-accessible, disabled, or blocked.*
>
> *In this paper we present 'GAIROSCOPE,' an ultrasonic covert channel that doesn't require a microphone on the receiving side. Our malware generates ultrasonic tones in the resonance frequencies of the MEMS gyroscope. [Micro-Electro-Mechanical System] These inaudible frequencies produce tiny mechanical oscillations within the smartphone's gyroscope, which can be demodulated into binary information. Notably, the gyroscope in smartphones is considered to be a 'safe' sensor that can be used legitimately from mobile apps and javascript. We introduce the adversarial attack model and present related work. We provide the relevant technical background and show the design and implementation of GAIROSCOPE.*
>
> *We present the evaluation results and discuss a set of countermeasures to this threat. Our experiments show that attackers can exfiltrate sensitive information from air-gapped computers to smartphones located a few meters away via a Speakers-to-Gyroscope covert channel.*

So this one is more serious and interesting. We were just talking about resonances last week with the Janet Jackson Rhythm Nation video and the Tacoma Narrows bridge. What's special about resonance is that a relatively small signal — like a gust of wind up the Tacoma Narrows — which would be entirely harmless in isolation, can sum into the power of successive properly-timed bits of energy to result in a significant signal. That's the effect that these guys have taken advantage of here.

They explain what's going on in the MEMS gyroscopes:

> *It is known that acoustic tones degrade MEMS sensors in a frequency range knowns as the 'resonance frequencies." This ultrasonic input produces erroneously low-frequency angular velocity readings in the X, Y, or Z direction(s). The vulnerability of MEMS sensors to ultrasonic corruption is due to the mechanical structure of a MEMS gyroscope. The misalignment between the driving and sensing axes is one of the main causes of the false output generated by the gyroscope. The phenomenon and its physical and mechanical roots are discussed in relevant literature. It was observed in the previous works that the typical resonance frequencies of MEMS are within a fragmented band in the ultrasonic frequencies range mainly above 18 kHz. The frequency of the resulting vibrations within the sensor is determined by the structure of the MEMS gyroscope, it's positioning, and the distance from the sound source.*

As always, they do all of the work of determining the natural resonance frequencies for a number of different Smartphones. And they demonstrate the ability to send 8 bits per second of binary data — completely covertly and silently — from a standard PC's speaker to a Smartphone located up to 8 meters away. 8 meters is a little over 26 feet away. That's pretty slick!

I’m always left thinking that it would be a real blaSt to be in this professor’s classes, being tasked with making these out-of-the-box attacks work. What fun that would be!