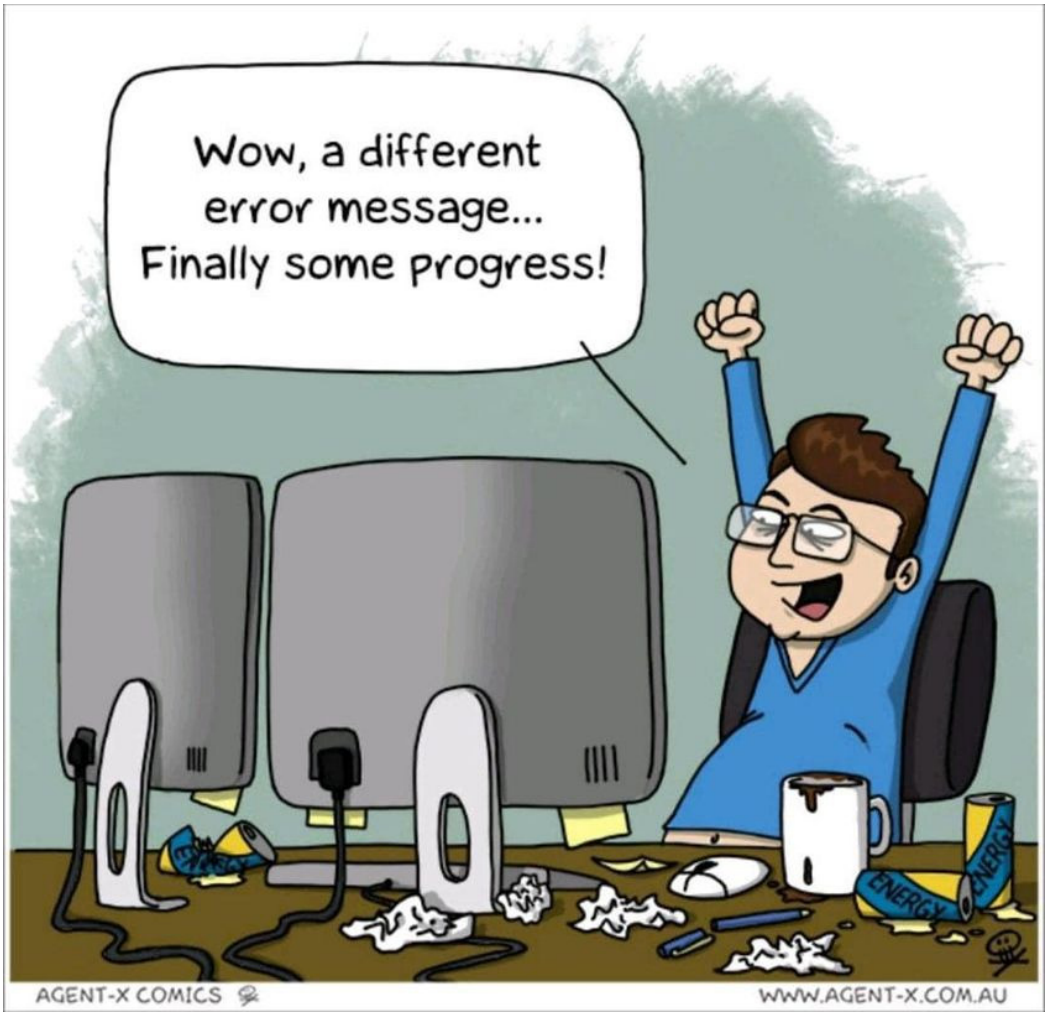
Security Now! #864 - 04-05-22

Port Knocking

This week on Security Now!

This week we examine a critical Java framework flaw that's been named "Spring4Shell" because it's mildly reminiscent of Java's recent "Log4J" problem. We'll also take a look at the popular QNAP NAS devices and several recent security troubles there. Sophos has got themselves an attention grabbing must patch now 9.8 CVSS vulnerability and it didn't take long (10-days) for the theoretical Browser-in-the-Browser spoof to become non-theoretical. There's more worrisome news on the NPM supply-chain package manager exploitation nightmare, the FinFisher spyware firm happily bites the dust, and some of the young hackers forming the Lapsus$ gang have been identified. Squarely in the doghouse this week is WYZE whose super-popular webcams have problems which are just as serious as those of the company itself... and, oh!, the authentication bypass details, which I'll share, are SO wonderful! Then after a bit of closing-the-loop feedback with our listeners, I want to talk about and put the idea of "Strong Service Concealment" on everyone's radar. "Port Knocking" is not a new idea by any means. But it is extremely clever, cool and useful. In today's world, there's more reason than ever for ports and the services behind them that are not actively soliciting public traffic to be kept completely hidden. There are a number of ways this can be done which are very cool.



0-Day Watch

Last week we noted Chrome's second 0-day of the year and titled the podast "Targeted Exploitation" with information thanks to Google's Thread Analysis Group research which documented the details of the exploitation of Chrome's first 0-day of the year.

This week we switch to Apple who, last Thursday, just pushed out patches for its own 4th and 5th 0-days of the year. Last year Google had a total of 16 for chrome and Apple was at 12. So for Apple to be at 5 for the first quarter suggests a run rate that might break both of their 2021 totals. Apple's 4th and 5th patches cover iPhones, iPad, and Macs. And these are true 0-day flaws since, in their typically obscure way, they said that the issues "may have been actively exploited." Uh huh... "may"?? So, why the big emergency?

In any event, we have an out-of-bounds write issue (CVE-2022-22674) in the Intel Graphics Driver that allows apps to read kernel memory — that's never what you want — and an out-of-bounds read issue (CVE-2022-22675) in the AppleAVD audio video media decoder that will enable apps to execute arbitrary code with kernel privileges. Both bugs were reported by anonymous researchers and resulted in iOS and iPadOS both moving to 15.4.1, and macOS Monterey to 12.3.1. The first flaw was fixed by improving input validation and the second by improving bounds checking.

Security News

Spring Forward *(Java: Spring4Shell)*

Another worrisome vulnerability in a Java framework has surfaced. The cybersecurity firm Praetorian said that the flaw impacts "Spring Core" on the Java Development Kit (JDK) versions 9 and later. It's a bypass for another much older vulnerability from way back in 2010, tracked as CVE-2010-1622 (ah, those quaint days when we only needed four digits to number our CVEs). If exploited, the bypass enables an unauthenticated attacker to execute arbitrary code on the target system, making it a remote code execution RCE. And unfortunately, a Chinese security researcher posted a working proof-of-concept (PoC) exploit to GitHub before deleting the account. But nothing remains hidden on the Internet so, indeed, the proof-of-concept code was quickly shared in other repositories and tested by security researchers, who confirmed it was a legitimate exploit for a new, potentially severe and previously unknown Java vulnerability.

"Spring" is a software framework for building Java applications, including web apps on top of the Java EE (Enterprise Edition) platform. Researchers said that "In certain configurations, exploitation of this issue is straightforward, as it only requires an attacker to send a crafted HTTP request to a vulnerable system. However, exploitation of different configurations will require the attacker to do additional research to find payloads that will be effective."

The Spring framework's maintainers, Spring.io, is a subsidiary of VMware. And last Friday they released emergency updates to fix the so-called "Spring4Shell" 0-day RCE vulnerability. I was trying to decide whether this would really be a 0-day if it wasn't being actively exploited in the wild against victims. It's a bit of a gray area. But I think that since public proof-of-concept exploits exist before a patch is ready, that probably qualifies it for 0-day status.

There was also some confusion because two other related vulnerabilities in the Spring Framework were also disclosed last week. There was the DoS vulnerability (CVE-2022-22950) and the Spring Cloud expression resource access vulnerability (CVE-2022-22963). But they're unrelated.

There has also been some questioning about just how bad this RCE really is. After their independent analysis, Flashpoint said "Current information suggests in order to exploit the vulnerability, attackers will have to locate and identify web app instances that actually use the DeserializationUtils, something already known by developers to be dangerous." And, of course we're talked a lot about the dangers of deserialization. Java is object oriented which means that an object can be a complex data structure. If one wants to store such a thing, the complex object is "serialized" into a byte stream for storage. And in order to later reconstitute that object it must be deserialized. And, a deserializer is inherently an interpreter. And naive interpreters are written to assume that they will only ever receive a valid serialization to deserialize.

The security firm Rapid7 said that despite the public availability of PoC exploits, "it's currently unclear which real-world applications use the vulnerable functionality." and "Configuration and JRE version may also be significant factors in exploitability and the likelihood of widespread exploitation."

However, CERT's CC vulnerability analyst Will Dormann tweeted that "The Spring4Shell exploit in the wild appears to work against the stock 'Handling Form Submission' sample code from spring.io. If the sample code is vulnerable, then I suspect there are indeed real-world apps out there that are vulnerable to RCE." And the flaw was assigned a CVE with a CVSS of 9.8 — so that's meant to grab everyone's attention. And, yesterday, VMware published security updates to remove the flaw from their Spring.io subsidiary's code.

But between the initial discovery of the vulnerability and yesterday's patch publication, exploitation of the vulnerability, where possible, did take off so much so that, as reflected by its 9.8 CVSS, it is now considered to be of critical importance to anyone using the Spring Framework because until it is updated any malicious actor with access to vulnerable applications can execute arbitrary commands and take complete control of a target system.

And, with communications and update delays always being possible, the widespread deployment of the Spring Framework for Java app development leads many security analysts to fear large scale attacks taking advantage of the so-called Spring4Shell vulnerability. The critical flaw impacts Spring MVC (Model-View-Controller) and Spring WebFlux apps running on JDK 9 and later.

QNAP and the OpenSSL DoS vulnerability

We recently talked about the denial-of-service bug that Tavis Ormandy and Chrome's TLS guy discovered in the OpenSSL library, which results in an infinite loop and processor capture when processing client certs using specific elliptic curve parameters.

Among the many companies whose products can likely be hung when they receive such a maliciously-crafted client cert is the Taiwanese company QNAP which last week revealed that a

selected number of its network-attached storage (NAS) appliances were vulnerable to this OpenSSL problem. Last Tuesday their advisory said: “An infinite loop vulnerability in OpenSSL has been reported to affect certain QNAP NAS. If exploited, the vulnerability allows attackers to conduct denial-of-service attacks.”

I have a list of the affected QTS and QuTS versions but QNAP doesn't yet have any patches available. The good news is, the only thing an attacker can do in this instance is temporarily lock-up your NAS. And they can only do that if TLS connections are being accepted from random IPs out on the public Internet. (And everyone knows that's NEVER a good idea, right?)

QNAP keeps being a mixed blessing. It's users generally love their devices despite them having had more than their share of security troubles so far just this year. QNAP is working to catch up to the OpenSSL DoS flaw while also working to patch the recent “Dirty Pipe” Linux kernel flaw from earlier in March which also currently has no mitigation on QNAP NAS devices. At least that one is only a local privilege-escalation vulnerability. And it's not QNAP's fault, any more than the OpenSSL problem is, since the trouble arises from the Linux kernel both problems share.

But, attackers have been pummeling QNAP devices all year with both ransomware and brute-force attacks to the point that the brute-force attacks prompted QNAP to urge its customers to remove their Internet-exposed NAS devices from the internet.

In late January, QNAP forced out an unexpected and not entirely welcome update to its customers’ NAS devices after warning them that the DeadBolt ransomware was mounting an offensive aimed at QNAP's users. And just two weeks ago reports surfaced that DeadBolt was at it again in a new wave of attacks against QNAP.

And last August, two vulnerabilities tracked as CVE-2021-3711 and CVE-2021-3712 that could result in remote-code execution (RCE) and denial-of-service respectively prompted emergency patches by QNAP.

This broader topic of the dangers of public port exposure serves as a perfect lead-in to today's discussion of Port Knocking, which is an extremely valuable, useful and clever solution that has not received the attention it's due. I think it's time to revisit the technology which is available to Linux users.

Sophos has a 9.8

Last week, the cybersecurity firm Sophos warned that a recently patched critical security vulnerability in its firewall product was now being actively exploited in real-world attacks. That flaw, CVE-2022-1040, sports an attention grabbing CVSS of 9.8 and it impacts Sophos Firewall versions 18.5 MR3 (v18.5.3) and earlier. What you never want to hear is that there's an authentication bypass vulnerability in the User Portal and Webadmin interface. And still worse, that when exploited it allows a remote attacker to execute code of their choosing.

The bad guys are aware of it. Sophos' security advisory stated: “Sophos has observed this vulnerability being used to target a small set of specific organizations primarily in the South Asia region. We have informed each of these organizations directly.”

And as for "doing it right" the flaw was addressed in a hotfix that is automatically installed for customers who have the "Allow automatic installation of hotfixes" setting enabled. I recognize that that whole issue of automatically pushing updates of security vulnerabilities is controversial. But at least in this instance, having Sophos taking responsibility for and maintaining their firewalls sure seems like a good idea. If it were mine, I'd be inclined to let Sophos autonomously maintain the firewall whose code they created. Not surprisingly, until the firewall is updated one way of the other, Sophos recommends that users disable WAN access to the User Portal and Web Admin interfaces. Yeah, no kidding.

And in a statement about both their integrity, their commitment to doing things right, and the severity of the issue, they have also provided updates to many earlier past-end-of-life versions of their firewalls and firmware. Sophos said: "Users of older versions of Sophos Firewall are required to upgrade to receive the latest protections and this fix."

CISA orders federal civilian agencies to patch the Sophos vulnerability

Last Thursday, the US CISA ordered federal civilian agencies to patch not only that critical Sophos firewall bug (presumably if they don't have auto update enabled) as well as 7 other vulnerabilities. And they have three weeks, or until April 21st, to get them all patched. All eight of these are under active exploitation.

In addition to the Sophos problem, the CISA also ordered federal agencies to patch a high severity arbitrary file upload vulnerability in the Trend Micro Apex Central product management console that can similarly be abused in remote code execution attacks. Two days earlier, Trend Micro said that it had observed "at least one active attempt of potential exploitation" of this vulnerability in the wild. The list of eight total "thou shalt patch" commandments include the need too patch a different QNAP NAS problem; an improper authorization vulnerability which has been reported to affect QNAP NAS running HBS 3 (Hybrid Backup Sync.) When exploited, the vulnerability allows remote attackers to log in to a device.

When looking over the list, I did a double take, since two of the eight CVEs are dated 2018 and one is from back in 2014. The two from 2018 are for Dasan GPON Routers. GPON stands for "Gigabit Passive Optical Network" and the routers suffer from a long-since-patched command injection vulnerability and authentication bypass vulnerability.

The CVE from 2014 is a bit startling. It's CVE-2014-6324 and its description in the National Vulnerabilities Database says: "The Kerberos Key Distribution Center (KDC) in Microsoft Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, and Windows Server 2012 Gold and R2 allows remote authenticated domain users to obtain domain administrator privileges via a forged signature in a ticket, as exploited in the wild in November 2014, aka Kerberos Checksum Vulnerability."

Believe it or not, eight years later, the US CISA feels the need to explicitly tell federal civilian agencies that they now have three weeks to get that patched. I'm sure all they needed was a gentle reminder.

Browser-in-the-browser

Two weeks ago, when we talked about the browser-in-the-browser attack during podcast #863, it was all just theoretical. Its penetration testing developer "Mrd0x_" had simply produced a proof of concept that demonstrated that pop-up windows could be made to appear indistinguishable from actual multi-factor authentication pop-ups. And in today's world it took less than 10 days for that "Hey! That's a great idea!" concept to become fully weaponized.

Last Thursday, a Belarusian threat actor known as Ghostwriter (aka UNC1151) has been spotted leveraging this recently disclosed browser-in-the-browser (BitB) technique as part of their credential phishing campaigns which are simultaneously exploiting the ongoing Russian invasion of Ukraine. As Mrd0x_ demonstrated, this technique allows a legitimate domain and pop-up to be shown to an unsuspecting user.

Google's Threat Analysis Group (TAG) write in a posting last Wednesday that Ghostwriter was using Mrd0x_'s browser-in-the-browser to siphon credentials entered by unsuspected victims.

In the TAG teams's posting they explained:

> *In early March, Google's Threat Analysis Group (TAG) published an update on the cyber activity it was tracking with regard to the war in Ukraine. Since our last update, TAG has observed a continuously growing number of threat actors using the war as a lure in phishing and malware campaigns. Government-backed actors from China, Iran, North Korea and Russia, as well as various unattributed groups, have used various Ukraine war-related themes in an effort to get targets to open malicious emails or click malicious links.*
>
> *Financially motivated and criminal actors are also using current events as a means for targeting users. For example, one actor is impersonating military personnel to extort money for rescuing relatives in Ukraine. TAG has also continued to observe multiple ransomware brokers continuing to operate in a business as usual sense.*

This should be no surprise. Anytime anything of importance happens anywhere, the scum surface in an attempt to leverage the event to their advantage, whatever it might be.

In this case Google's group wrote that: "Ghostwriter actors have quickly adopted this new technique, combining it with a previously observed technique, hosting credential phishing landing pages on compromised sites."

So, here we have an example of a purely theoretical proof of concept being picked up within days of its publication and being leveraged to significantly increase the effectiveness of a traditional phishing and logon campaign.

The supply-chain attacks on NPM have been growing

When we first talked about this last week, the security firm JFrog had identified a total of 218 malicious packages which were using a form of name collision to replace packages in the @azure namespace. By naming their malicious packages without any namespace designation, their packages might be obtained when a developer had not explicitly specified the @azure namespace for the target.

At the time, JFrog had not identified the threat actor behind this NPM repository attack. Now, a week later we know more.

The threat actor named “RED-LILI” has been linked to this ongoing large-scale supply chain attack campaign targeting the NPM package repository and has published nearly 800 malicious modules.

The Israeli security company Checkmarx said: “Customarily, attackers use an anonymous disposable NPM account from which they launch their attacks. But this time, the attacker has fully-automated the process of NPM account creation and has opened dedicated accounts, one new account per package, thus making his new malicious packages much more difficult to spot.”

Checkmarx's findings build upon recent reports from JFrog and Sonatype, which detailed hundreds of NPM packages which leveraged the dependency confusion typosquatting style package replacement to target Azure, Uber, and Airbnb developers.

According to a detailed analysis of RED-LILI's modus operandi, earliest evidence of anomalous activity was found to have occurred on February 23, with the cluster of malicious packages published in "bursts" over a span of a week.

Specifically, the automation process for uploading the rogue libraries to NPM, which Checkmarx described as a “factory,” involves using a combination of custom Python code and web testing tools like Selenium to simulate user actions required for replicating the user creation process in the registry.

Bypassing the one-time password (OTP) verification barrier put in place by NPM is no problem since NPM sends a one-time password to the eMail address the attacker's Bot registers with. So the NPM account creation automation simply reads the eMail and responds with the eMailed OPT. Then, equipped with this brand new NPM user account, the threat actor creates and publishes a malicious package — one per account — using automation, but not before generating an access token to enable it to publish the package without requiring an email OTP challenge.

The Checkmarx researchers said: “As supply chain attackers improve their skills and make life harder for their defenders, this attack marks another milestone in their progress. By distributing the packages across multiple usernames, the attacker makes it harder for defenders to correlate and take them all down with 'one stroke' as had traditionally been possible.

So, I read this as the chickens coming home to roost. The NPM system never had super-tight security. And that was fine for a while. But now it's not. And its lack of tight security is finally becoming a problem. The only way to combat this will be to impose much more stringent strictures on account creation and content publication. The bad guys are despoiling everything they can get their hand on.

FinFisher bites the dust

FinFisher has been lurking around as one of the more successful and prevalent commercial spyware purveyors. The good news is that the Munich Germany-based spyware company

formally declared its insolvency last month amid an ongoing and certainly unwelcome investigation into its business dealings.

They made the mistake of selling their premiere spyware product "FinSpy," to the Turkish government — without having the legal documentation to do so — after which their FinSpy system was used in a Turkish operation that preyed on anti-government protestors. We talked about this at the time. Legal complaints filed by Reporters Without Borders, Netzpolitik, the Society for Civil Rights, and the European Center for Constitutional and Human Rights accused FinFisher of failing to abide by European export regulations including the requirement to obtain a permit granting trade to non-EU countries by the Federal Office of Economics and Export Control (BAFA). FinSpy was created in 2016 and has been linked to customers including the governments of Egypt, Bahrain, Bangladesh, Ethiopia, Oman, Saudi Arabia, and Venezuela.

According to the NGO's investigation, "There are urgent indications that the Munich-based company conglomerate sold the spy software FinSpy to the Turkish government without the approval of the federal government and thus contributed to the surveillance of opposition figures and journalists in Turkey." So, about a year and a half ago in October 2020, German authorities raided FinFisher's corporate offices, two associated businesses, and the residencies of directors and executives — leading to the recent announcement that FinFisher accounts were seized and operations halted. So, that's likely the end of that operation.

A LAPSUS$ in judgment

Recall that three weeks ago, during our QWACs on, QWACs off episode, we mentioned the attack on NVIDIA's networks and the attacker's subsequent exfiltration of 1TB of NVIDIA's data which included some expired NVIDIA driver signing certificates. Those certificates were then immediately used to sign malware, and I was puzzled over how and why Windows would choose to honor drivers signed by certificates that were expired at the time of their signing. That mystery remains. The mystery that no longer remains regards a couple of the perpetrators behind that and apparently many other recent very high profile attacks including the likes of Microsoft, Nvidia, Samsung, Okta, and Ubisoft with many of them resulting in massive data leaks.

The group calls itself Lapsus$ — spelled L A P S U S with an additional trailing dollar sign. And despite the trailing dollar sign and their high-profile victim list, most Lapsus$ members are believed to be teenagers driven mainly by their goal of making a name on the hacking scene and not by financial motivation. And if making a name for themselves on the hacking scene was their goal, then mission accomplished!

Two members of the group, teenagers aged 16 and 17, were in the news last week when they were arrested and arraigned, appearing at the Highbury Corner youth court, in London, last Friday, charged with a number of cyber offenses. The names of both young men, being minors, are being kept private and both were released on bail. They've both been charged with three counts of unauthorized access with intent to impair operation of, or hinder access to a computer, and two counts of fraud by false representation. Additionally, the 16-year-old has also been charged with one count of causing a computer to perform a function to secure unauthorized access to a program.

And this pair appears to be part of a larger group, because also last week the City of London Police, which is leading the international investigation into Lapsus$, announced that it had arrested 7 people, all between the ages of 16 and 21, in the UK alone.

In the US, our FBI is looking into the group's illegal activities and is seeking info concerning the Lapsus$ members involved in the compromise of computer networks belonging to multiple US-based companies. The FBI said: "These unidentified individuals took credit for both the theft and dissemination of proprietary data that they claim to have illegally obtained. The FBI is seeking information regarding the identities of the individuals responsible for these cyber intrusions."

While it's still unclear how many active members the gang has and what roles each of them plays, based on Telegram chats it is believed that they at least have affiliates located all over the world speaking multiple languages, including English, Russian, Turkish, German, and Portuguese.

Their bail and release was said to have "conditions" <unquote>. Wanna bet that one of those conditions is an utter and total parental-enforced ban from any use of an internet-connected device? If so, that might explain why around the time of the news of the arrests, Lapsus$ told its nearly 58,000 Telegram followers that some of its members would be taking "a vacation."

But those recent arrests haven't put a dampener on the larger group's activities because last week 70GB of data belonging to the software services giant "Globant" were leaked on March 30. Globant, with headquarters in Luxembourg, said that they're currently "conducting an exhaustive investigation" and that it's "taking strict measures to prevent further incidents." I'll bet they are.

Not so Wyze

One week ago, last Tuesday, BitDefender published the results of their close examination of the very popular WYZE family of security and surveillance oriented Internet connected web cams. And it will surprise no one to learn that they found problems, nor that the problems were extremely critical given the application these webcams are typically deployed for. And I utterly LOVE the details of the authentication bypass hack they found, which I'll describe in a minute.

But the most distressing aspect of the story is the fact that the BitDefender group has been working with Wyze, or perhaps better stated "attempting to work with Wyze" for THREE YEARS to get three critical problems resolved.

- Mar 06, 2019: Bitdefender makes first contact with vendor and asks PGP key via support form.
- Mar 15, 2019: Bitdefender makes a second attempt at getting in touch with the vendor, still without response.
- April 22, 2019 - Wyze releases updates for Wyze Cam v2 in v4.9.4.37 and reduces risk for unauthenticated access to the contents of the SD card. Still no contact with our research team.
- April 23, 2019 - v4.10.3.50 released for Wyze Cam Pan v1 with the same risk reduction for unauthenticated access to the contents of the SD card.
- May 22, 2019: Bitdefender reserves CVE number pending publication

- September 24, 2019 - Update released for Wyze Cam v2 that fixes the CVE-2019-9564 issue.
- November 9, 2020 - Vendor fixes the CVE-2019-12266 issue with an app update.
- Nov 10, 2020: Vendor acknowledges reception and assigns an internal contact
- Nov 12, 2020: Advisory and proof of concept are shared with the vendor
  < 9 months pass >
- Aug 31, 2021: Bitdefender follows up on patch progress
- Sep 13, 2021: Bitdefender notifies vendor of upcoming publication
  < 4 and a half months pass >
- Jan 29, 2022: Vendor releases firmware update to fix the unauthenticated access to the contents of the SD card issue.
  < BitDefender still patiently waits 60 days from January 29th and on March 29th… >
- Mar 29, 2022: Bitdefender publishes this report

I've said it before, and I'm quite sure this won't be the last time that I say it again: There is something very wrong with the idea that an independent security research group must expend this level of effort to not only reverse engineer and examine a product whose security is critically important to its users, but to then face an utterly unresponsive product publisher and attempt, for three years, to get them to fix critical flaws in the operation of their surveillance webcams.

And look at the Catch-22 that BitDefender is in. The only way to leverage responsibility from Wyze would be to go public with the news and details of the flaws. But doing so would immediately place all of Wyze's gazillion webcam users at significant risk. And even if details were withheld, we've seen many instances where just telling the bad guys where to look for vulnerabilities is all that's necessary. Those wearing black hats could certainly follow in BitDefender's footsteps. So BitDefender had little true choice other than to wait and push and poke and prod and hope that Wyze would eventually open a responsible dialog.

And it's just rich that Wyze's cybersecurity team finally said they appreciated the responsible disclosure provided by Bitdefender on the vulnerabilities. Yeah, I bet they did. Because of BitDefender's ethics, Wyze had BitDefender over a barrel.

Okay. So get a load of this amazing remote connection authentication bypass. It's just the best thing ever! ...

When connecting remotely, a client is required to log onto the device. The client and the webcam share a 128-bit secret key. So the client initiates its connection by sending an IOCTL command with the ID 0x**2710**. The Wyze cam generates a random 'nonce' value which it encrypts with the 128-bit shared secret key. It sends the encrypted blob to the client. By the design of this simple protocol, the client must have the same 128-bit shared secret key. It uses that key to decrypt the camera's randomly chosen 'nonce' value and then authenticates itself to the camera by returning the properly decrypted camera nonce on an IOCTL command with the ID 0x**2712**.

The camera, receiving the 0x**2712** IOCTL, compares the nonce that was hopefully decrypted by the connecting-client to the value that it stored locally and only if they match will the authentication succeed and the connection be accepted. And after that the client is free to do whatever it wishes with the camera.

But here's what the BitDefender guys found. And it's so classic: The way the Wyze firmware works is that upon receiving the initial 0x**2710** command, it generates and stores the nonce for subsequent comparison... which it then encrypts and sends to the client.

But, if the client NEVER sends the 0x**2710** command in the first place, the nonce's stored value remains set to **all 0's**. So all any attacker needs to do, to gain full access to any original or still unpatched Wyze cam, is to connect and simply skip issuing the first 0x**2710** command which asks the camera to begin the authentication handshake. An attacker simply first sends the second 0x**2712** command with an **all-0's** authentication. Since that will always match the camera's default null nonce, **anyone** can log into **anyone's** Wyze cam.

BitDefender wrote: *"After authentication, we can fully control the device, including motion control (pan/tilt), disabling recording to SD, turning the camera on/off, among other things. We cannot view the live audio and video feed, though, because it is encrypted under that same still-unknown shared private key. However, we can bypass this restriction by daisy-chaining a stack buffer overflow which leads to remote code execution (RCE) as detailed in Part 2."*

They wrote: *"For the stack buffer overflow, when processing IOCTL with ID 0x2776, the device does not check whether the destination buffer is long enough before copying the contents onto the stack. Exploiting this vulnerability is straight-forward. Through the IOCTL with ID 0x**2776** we can set which servers to use to connect to the cloud. This seems to be a debugging function that allows for the selection of production, beta or internal API servers. When sending a request, we specify the length of the buffer in the first byte, then the buffer itself. This content is then copied onto the stack into a fixed 0x40 (64 byte) length buffer. Even though the specified size in the first byte is taken as signed INT, a size of 0x7F (255) is enough to overwrite the return address of the function and run attacker-provided code."*

The third and final flaw is unauthenticated access to the contents of the SD card:

When inserting an SD card into the camera, the contents of the SD card (including the recordings) can be accessed via the web server listening on port 80 without authentication. This is enabled by the fact that, after an SD card is inserted, a symlink to the card mount directory is automatically created in the www directory, which is served by the webserver. The card contents can be viewed through the hello.cgi functionality located at /cgi-bin/hello.cgi; then the files can be downloaded through the /SDPath/ path. The SD card also holds the camera's log files. Before writing them to the card, the device XORs the content with 0x90 — not very strong protection. These log files can contain sensitive info, such as "UID" and the shared private key, which can then be used to connect remotely.

The good news is that the 2nd and 3rd generation Wyze cams can be updated to cure these various problems. The bad news is that the 1st generation cameras have been abandoned and Wyze has said that they do not plan to support or update them. Perhaps all of the press that this is now, finally, after three years, generating with the negative publicity of having critically broken and trivial-to-hack 1st-generation webcams might cause them to change their minds. But unless they do, anyone using a 1st generation Wyze cam is strongly encouraged not to expose it to any possible hacking. And anyone with 2nd or 3rd generation cams should update everything immediately if not sooner.

Closing The Loop

The Nargles / @WithTheNargles
Thanks to your recommendation, I just finished the first Bobiverse book. All I can say is, thanks for the recommendation - it was a lot of fun! And Ray Porter's narration (particularly of GUPPI) is spot on. @SGgrc @leolaporte

Scott Cleveland / @clevelandscott
@SGgrc A few weeks ago you and Leo were talking about the Bobiverse we are legion (we are Bob) books… Thank you! It's so hard to narrow down when scrolling through Audible what I will like. Your suggestions are pretty much a spot on automatic win for me.

If the quality of my recommendations is to hold, I should probably note that book #4 IS dragging noticeably. It's still okay, but Dennis appears to be running out of new ideas for his Bobs. He's reusing the ideas he has, and he has built an extremely interesting and clever Bobiverse using subspace comm-links and virtual reality in clever and feasible ways, given everything else he's constructed. But perhaps it should have been left as a trilogy.

David Lemire / @dlemire60
Hi Steve. I recently bumped into the author of NoScript on Twitter (when I mentioned I'd abandoned it long ago). He encouraged me to take a fresh look, so I found the website. Looking at the usage page it does appear that the program has been updated / adjusted to the realities of the modern web. I know you also gave up on it long ago, but this was interesting enough I thought maybe it was time for a revisit of NoScript. And no, I haven't actually tried playing with it myself yet, or I'd include my experience here.

I appreciate knowing that NoScript hasn't thrown in the towel yet, despite its name. The author is probably suffering the same dilemma I am with SpinRite. Renaming his program "SomeScript" doesn't really pack the same punch. Our development group's discoveries with SpinRite strongly indicate that SpinRite's future with solid state mass storage is guaranteed, perhaps maybe even more so than it was with spinning media. But I can't change the name even though someday nothing will be spinning anymore.

Port Knocking

"knockd"

OpenWRT / DD-WRT /

Linux Magazine: Remote access security with single-packet port knocking
https://www.linux-magazine.com/Issues/2008/91/Single-Packet-Port-Knocking

How To Knock Into Your Network (DD-WRT)
https://www.howtogeek.com/104548/how-to-knock-into-your-network-dd-wrt/

How to Knock into Your Network, Part 2: Protect Your VPN (DD-WRT)
https://www.howtogeek.com/105693/how-to-knock-into-your-network-part-2-protect-your-vpn-dd-wrt/

Single-Packet Authorization

Fwknop - "FireWall KNock OPerator"
https://www.cipherdyne.org/fwknop/

Single Packet Authorization: A Comprehensive Guide to Strong Service Concealment with fwknop
https://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html

Component / Technology	Fedora/RHEL/CentOS	Debian/Ubuntu	OpenWRT	FreeBSD	Mac OS X	OpenBSD	iPhone	Android	Cygwin	Windows
fwknop client	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
fwknopd server	Y	Y	Y	Y	Y	Y	N	N	N	N
GnuPG Support	Y	Y	Y	Y	Y	Y	N	N	N	N
HMAC Support	Y	Y	Y	Y	Y	Y	N	Y	Y	N
Client NAT Access Support	Y	Y	Y	N	N	N	N	Y	Y	N
fwknopd Server-Side NAT	Y	Y	Y	N	N	N	N/A	N/A	N/A	N
Native Windows binary (client)	N/A									Y
libfko library	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
perl libfko bindings	Y	Y	Y	Y	Y	Y	N	N	Unknown	Unknown
python libfko bindings	Y	Y	Y	Y	Y	Y	N	N	Unknown	Unknown

https://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html