Security Analysis of Smart connected Cameras: Case Study on the Eufy Doorbell

Kushal Goenka
University of Illinois at Urbana-Champaign
Champaign, IL
kgoenka2@illinois.edu

Kevyn Cheng
University of Illinois at Urbana-Champaign
Urbana, IL
kkcheng2@illinois.edu

***Abstract*—In this project we aim to determine if and when manufacturers patch their IoT devices when vulnerabilities for similar products are discovered. To approach this on a small scale, we will attempt to examine and potentially expose vulnerabilities in either the Device, Mobile Application, Cloud Endpoints, or the Network Communication of the smart-home security system which incorporates smart doorbell cameras from Eufy. Our threat model will look at three types of attackers, an Off-Path Attacker who can exploit vulnerabilities in the mobile application or the cloud endpoints, an On-Path attacker who is already on the user's local network, or an attacker who is geographically close to the victim but not on the actual network [1].**

To streamline our approach, we will first examine inter-company vulnerabilities which exist, from vulnerabilities found in other similar/competing products such as the Ring Doorbell camera, or similar smart home devices such as the Nest indoor security cameras. We believe that this will provide a good framework to look at existing vulnerabilities, and if they have indeed been patched by these manufacturers. We will look at the Common Vulnerabilities and Exposures (CVE's) list, which provides a list of publicly disclosed vulnerabilities and exposures. An example of the same is CVE-2019-9483 which allowed attackers to obtain video and audio data, or insert spoofed video that did not correspond to the actual person at the door [2].

***Index Terms*—IoT, CVE, vulnerabilities, threat model, smart doorbell, Eufy**

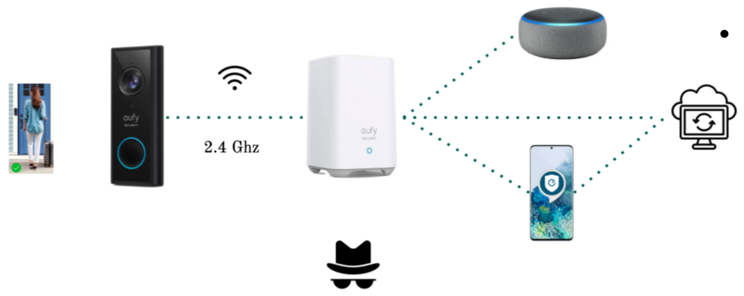
I. Introduction

Taking a look at the current overall system, it becomes clear that there are a range of potential attack surfaces which can be exploited by attackers. One version of the device, the battery powered doorbell camera works by first detecting an individual in front of the camera. This is done via a touted on device AI mechanism which prevents the data from being sent to any servers for processing. The device then communicates with the Homebase via WiFi (2.4 Ghz band), which is able to store the data on the device, store it in the cloud if a subscription service has been opted into, or stream the data after a certain delay to a mobile application. The Homebase is connected to the network via an Ethernet cable, and although unclear, looking into the documentation reveals that there could also be some amount of interaction, between the Homebase and Doorbell camera, happening over Bluetooth to prolong battery.

The setup process is straightforward with the homebase pairing with the camera via a "chime" that is played. The camera is said to have about 180 day charge, with 10 minutes of usage everyday on average, or may be connected to existing doorbell wiring for an always powered device.

In this paper, we specifically look at this company, along with this specific type of IoT device as we believe it has a large attack surface and presents many other security vulnerabilities for the entire home security ecosystem. Doorbells are the first line of defense against intruders and burglars and provide a compelling reason for many to find and exploit security vulnerabilities in them. They can act as a deterrent to potential crimes, and hence need to be as secure as possible. With the increase in popularity of these home security systems, we have seen greater communication between devices where doorbell cameras are connected to a smart-lock to remotely allow access for guests and authorized users. Amazon, along with its Ring security systems has introduced "Amazon Key", a service to allow delivery drivers to deliver packages inside the home, rather than on porches where they might be stolen. Since such a device is generally placed outside the home, its easier for an attacker geographically close to the victim to eavesdrop, pretend to be a Man in the Middle (MITM), or sniff on packets. This happens because a poorly secured WiFi network could allow attackers to be on path and inject traffic such as spoofed packets and deauthentication packets, and a poorly secured router could then allow the attacker to become a MITM attacker. Vulnerabilities in these devices can then compromise the entire home security ecosystem and any form of data transmission from a previously secured internal network can be exposed.

Furthermore, we decided to place our efforts into vulnerabilities in the Eufy Doorbell camera for two main reasons. In literature we failed to find any references to Eufy's smart home devices. This could be a factor of either their devices being extremely secure, or a lack of research having been conducted to find exploits. Through this paper we hope to at least present a baseline for which attacks this security system is resilient against. Secondly, Eufy is a new competitor to established firms such as Nest, Ring, Arlo and August.



II. Background

This project intends to target consumer-grade IoT devices, particularly Cameras and Doorbell cameras that have become increasingly popular. Touted as home security devices, there have been many instances where these devices have been found to have serious vulnerabilities as well. The market for such doorbell cameras in 2020 is expected to be about $1.83 billion in the US alone [4]. Looking at various popular devices on the market, we chose to take a deeper look into the wired, as well as the battery powered Eufy Doorbell camera.

The world of the Internet of Things is quite expansive and covers everything from edge devices to connectivity devices, fog computing, data accumulation devices, application, and processing [5]. For the purposes of this paper, we will only be focused on edge devices that gather and send data. As consumers purchase more and more "smart" IoT devices, they are filling their home with devices that frequently focus more on features and inter-connectivity over security. Examples of such devices include smart TVs, appliances, lights, security cameras, door locks, or smart home hubs like Amazon Alexa. Because these devices typically are connected to the Internet, the attack surface of a home network increases substantially, and many of these devices cease to function properly if internet access is blocked.

Risk factors include both loss of privacy, complicity in a botnet, or safety concerns. For example, a baby monitor video device can be compromised, and an attacker can monitor the baby and speak to it. Many IoT devices are also not patched by the manufacturer, and thus are susceptible to being compromised and used in a botnet to be used in DDoS attacks.

III. Related Works

This paper's background is based off several papers detailing various attacks on IoT as well as CVEs covering known vulnerabilities in IoT devices. The Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic by Reisman et al [7] details how even encrypted IoT traffic can allow an attacker to learn sensitive information about users and their home. The Mirai botnet and the IoT Zombie Armies by G. Kambourakis et al presents an overview of the target rich environment of IoT devices and a case study of a IoT botnet [3]. Dolphin Attack: Inaudible Voice Commands by Zhang et al covers inaudible, ultrasonic audio signals that are interpreted by voice recognition controlled devices as valid commands.

A. Risk Factors and Related Works

- Privacy: With always on devices that can monitor aspects of user life, through cameras, microphones, and other sensors, privacy is at risk both from the service provider and attackers [7]. For example, a camera that detects motion might start transmitting footage, or a smart lock can send an update message when it's status is changed. Even with devices that communicate over encrypted channels, a network observer such as an ISP can infer metadata and know that some action is happening. This can allow an attacker to learn the movements of people in their home as well as ease of physically breaking in. Another example is sleep tracking devices. If data is not transmitted securely, an attacker can gain very sensitive data about a user's home and also their health.
- Botnet: Many IoT devices are cheap, insecure devices without consistent access to security updates. Manufacturers have little financial incentive to provide patches even for known vulnerabilities and end users often do not change the default username and password. In addition, efficient network scanning and simple botnet behavior make it easy to find vast amounts of vulnerable devices that can run simple code. These factors allow malware such as the original Mirai to easily compromise a few hundred thousand devices to use as a DDoS botnet [3]. Mirai starts by scanning Telnet ports and attempting to establish a connection with common credentials. Upon successful connection, server sends a loader to deploy a architecture-specific, malware binary that listens for commands from the command and control server. With the release of the Mirai's source code, subsequent variations continue to target IoT devices today.
- Physical safety: The Dolphin Attack has shown that speech recognition software can be targeted by ultrasonic frequencies that humans cannot hear, but is picked up by the device [6]. This can be used to issue commands to devices such as smart home bases to open doors, car navigation systems to redirect the driver, and smartphone assistants to make calls.

These attacks can be combined to pull off attacks on IoT devices that give the attacker an extreme amount of flexibility. For example, an attacker can compromise a device with a speaker, such as the camera systems we are targeting, and output a command to a smart home devices that would unlock a smart lock on a door. Additionally, an attacker could determine the location of a home's occupants either through traffic analysis or compromising an IoT device to carry out a physical attack on the location.

B. Known CVEs

Given that we wish to first examine and test known Common Vulnerabilities and Exposures on the device(s) in question, we would like to enumerate some of those as these are relevant and have a high risk of being discovered and exploited in other similar devices as well.

- CVE-2019-9483: Amazon Ring Doorbell before 3.4.7 mishandles encryption, which allows attackers to obtain audio and video data, or insert spoofed video that does not correspond to the actual person at the door [8].
- CVE-2019-3984: Blink XT2 Sync Module firmware prior to 2.13.11 allows remote attackers to execute arbitrary commands on the device due to improperly sanitized input when the device retrieves updates scripts from the internet [9].
- CVE-2019-3950: Arlo Basestation firmware 1.12.0.1_27940 and prior contain a hard coded username and password combination that allows root access to the device when an onboard serial interface is connected to [10].
- CVE-2019-3949: Arlo Basestation firmware 1.12.0.1_27940 and prior firmware contain a networking misconfiguration that allows access to restricted network interfaces. This could allow an attacker to upload or download arbitrary files and possibly execute malicious code on the device [11].
- CVE-2019-5043: An exploitable denial-of-service vulnerability exists in the Weave daemon of the Nest Cam IQ Indoor, version 4620002. A set of TCP connections can cause unrestricted resource allocation, resulting in a denial of service. An attacker can connect multiple times to trigger this vulnerability [12].
- CVE-2019-3988: Blink XT2 Sync Module firmware prior to 2.13.11 allows remote attackers to execute arbitrary commands on the device due to improperly sanitized input when configuring the devices WiFi configuration via the BSSID parameter [13].
- CVE-2015-4400: Ring (formerly DoorBot) video doorbells allow remote attackers to obtain sensitive information about the wireless network configuration by pressing the set up button and leveraging an API in the GainSpan Wi-Fi module [14].

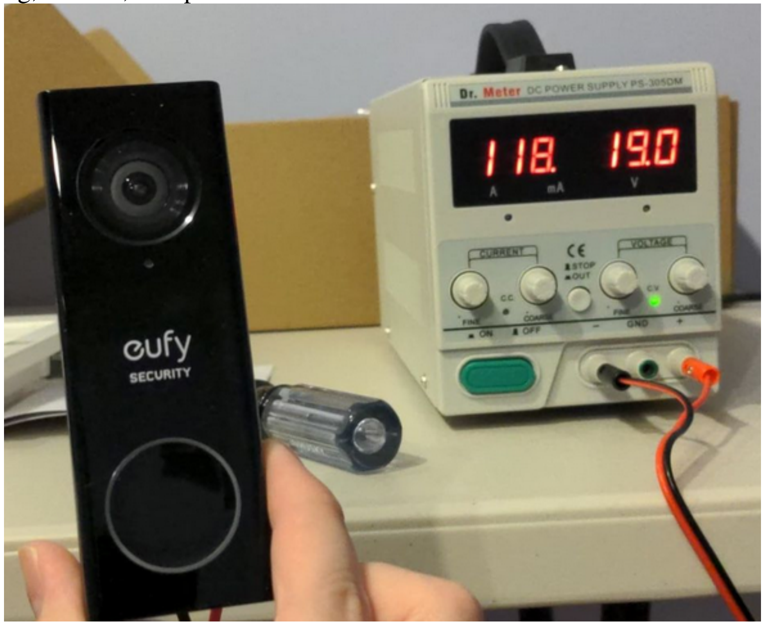
In addition to this, we found references to more vulnerabilities. These were found in news articles and other disclosures and were not attributed to any particular CVE.

- March 2017: Ring doorbell sent packets of Audio data to servers in China [15].
- May 2018: Ring app doesn't immediately revoke access when an account password changes [16].
- Nov 2019: On setup, the Ring Doorbell sent a user's WiFi password in plaintext to the device, along with a vulnerability forcing a user to reset the device [17].
- August 2020: August 2020 - Eufy T8200 video doorbell, exposed account information, such as email addresses and WiFi passwords [18].

IV. Experimental Setup

We experimented on both the Eufy Wired Doorbell camera and the Eufy Wireless Video Doorbell. The wired camera is always on, and does not have a base station. It does have a standalone chime that didn't seem to function properly, and is not required for the operation of the doorbell. It can accept either 16-24 volt of AC power at 0.3A or 19V at about .6A although it usually uses a lot less amperage. It has two screws in the back, where you are supposed to wind AC power cables around. For this experiment, we simply clipped alligator clips on the screws and secured it with tape. It features both wifi and bluetooth connectivity. The bluetooth is used to send control messages from the phone app to choose the wifi network to connect to and to reboot the device. All other messages appear to go through WiFi. In order to capture packets accurately using Wireshark, we connected the doorbell to a hotspot hosted on a Dell Precision 5510 running Linux Mint.

To conduct our experiments on the Wireless Doorbell Camera, our hardware consisted of a Wireless network card, TL-WDN3200. We further used Wireshark to capture and analyze TCP/UDP packets and DNS Queries. We used multiple operating systems including Linux Mint OS, Ubuntu 14.04, and Kali Linux. Furthermore, we also leveraged tools such as aircrack-ng, Nessus, nmap and binwalk.



The model number of the Battery Powered Doorbell is T8210, running the latest System Version 2.2.2.1. The base station was the Homebase 2, running a System Version 2.1.3.3h and Subsystem version 1.3.0.9.

The model number of the Wired version is T8201, running the latest system version 2.328.This version was published on 11/28/2020, which shows that the manufacturer has been issuing system updates, which is a very good sign.

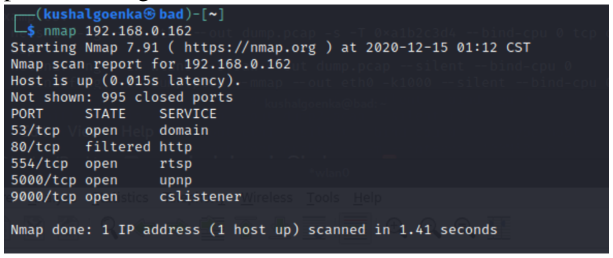
V. Experiments Conducted

A. Port Scan

We know from prior vulnerabilities and general network security principles that having open ports can result in vulnerabilities in devices that can be exploited. There has been some research to show that more the number of open ports on a networked device, there is a greater chance that it will be susceptible to attacks. We conducted preliminary port scans via nmap to determine which ports on the Eufy doorbell devices were open. Looking at other devices such as smart switches, smart speakers, streaming devices, we had noticed that as a general rule of thumb, the ports were either closed or filtered.

On the Wired Doorbell, we saw that all ports appeared to be closed and this is a good sign that the Eufy is taking proactive steps to prevent any possible adversarial attacks as a result of open ports on the network.

However, when analyzing the ports on the Wireless Doorbell, we noticed that given the additional functionality built into the device, there were open ports, as seen in Table 1. The ports that were open are Port 53, 554, 5000, and 9000, with port 80 being filtered.



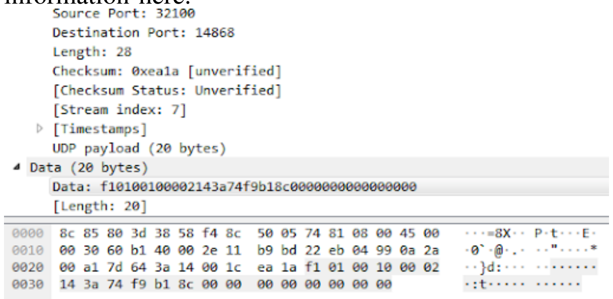
```
┌──(kushalgoenka㉿bad)-[~]
└─$ nmap 192.168.0.162
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-15 01:12 CST
Nmap scan report for 192.168.0.162
Host is up (0.015s latency).
Not shown: 995 closed ports
PORT     STATE    SERVICE
53/tcp   open     domain
80/tcp   filtered http
554/tcp  open     rtsp
5000/tcp open     upnp
9000/tcp open     cslistener

Nmap done: 1 IP address (1 host up) scanned in 1.41 seconds
```

One point of concern was that port 554, reserved for RTSP is open when the feature isn't available on the Eufy doorbells. Exploring

B. *Traffic Analysis*

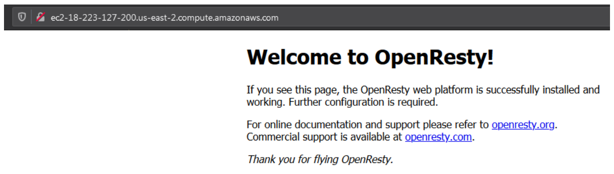
First, we started Wireshark, and then plugged the power into the wired doorbell in order to capture all of its packets sent and received. The first interesting event that happens after DHCP is a DNS request for time.nist.gov, and then gets the time from using the NTP protocol. Afterwards, it makes connections to 54.153.101.7, 18.223.127.200, 34.235.4.153 which are all amazon ec2 domains and gets the same response from each. F10100100002143a74f9b18c0000000000000000. We assume this value is a status message that the doorbell is able to interpret. Unfortunately, the human readable value is just blank characters with a t, so we are not able to find meaningful information here.



```
    Source Port: 32100
    Destination Port: 14868
    Length: 28
    Checksum: 0xea1a [unverified]
    [Checksum Status: Unverified]
    [Stream index: 7]
  ▷ [Timestamps]
    UDP payload (20 bytes)
◢ Data (20 bytes)
    Data: f10100100002143a74f9b18c0000000000000000
    [Length: 20]

0000  8c 85 80 3d 38 58 f4 8c  50 05 74 81 08 00 45 00   ···=8X·· P·t···E·
0010  00 30 60 b1 40 00 2e 11  b9 bd 22 eb 04 99 0a 2a   ·0`·@·.· ··"····*
0020  00 a1 7d 64 3a 14 00 1c  ea 1a f1 01 00 10 00 02   ··}d:··· ········
0030  14 3a 74 f9 b1 8c 00 00  00 00 00 00 00 00         ·:t····· ······
```

A reverse DNS lookup yields the CNAMEs: ec2-54-153-101-7.us-west-1.compute.amazonaws.com, ec2-18-223-127-200.us-east-2.compute.amazonaws.com, and ec2-34-235-4-153.compute-1.amazonaws.com. Visiting any of these addresses displays the landing page for OpenResty, which is a web platform based on nginx, that runs Lua scripts.Without probing the EC2 instances, this is as far as we are exploring this angle.



ec2-18-223-127-200.us-east-2.compute.amazonaws.com

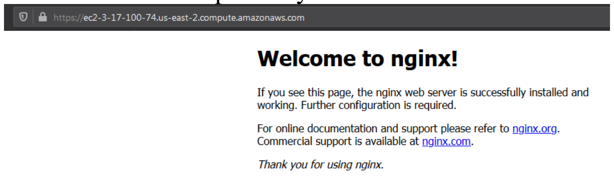
Welcome to OpenResty!

If you see this page, the OpenResty web platform is successfully installed and working. Further configuration is required.

For online documentation and support please refer to openresty.org.
Commercial support is available at openresty.com.

Thank you for flying OpenResty.

It then makes a DNS query for security-app.eufylife.com, which redirects to 3.17.100.74, another amazon ec2 domain. Ec2-3-17-100-74.us-east-2.compute.amazonaws.com. This the server it establishes a connection with. It will continue to make this DNS query and connect to other amazon ec2 instances. Visiting this site takes us to a nginx landing page, which only tells us that this is probably a webserver.



https://ec2-3-17-100-74.us-east-2.compute.amazonaws.com

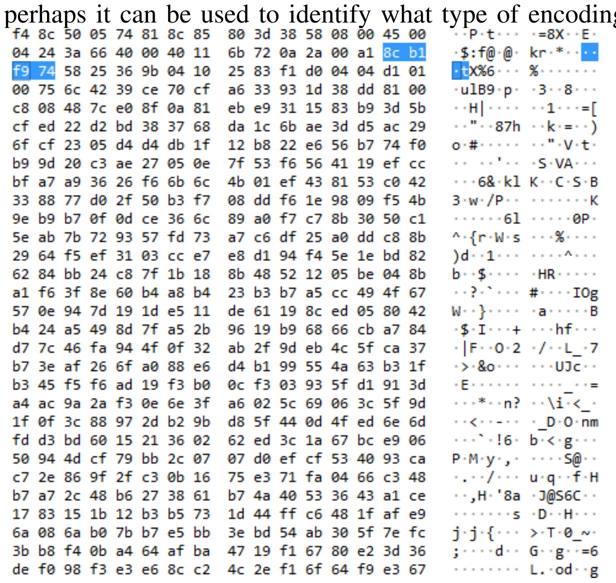
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

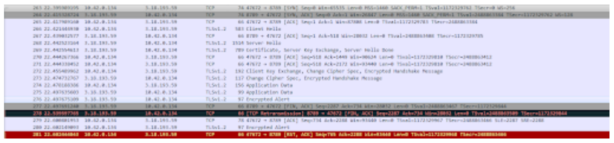
Thank you for using nginx.

When you have the Eufy app open on your phone, the doorbell camera will send short packets to both your phone and your public IP address, which isn't helpful in tracking where the packet is going because the NAT is sending it to the real destination. The length of the payload is 24 bytes, When video is streaming, the packets are length 1032, so it's obvious when video is being transmitted. The packets are being sent to your phone's IP address, so that is quite straightforward. On a static image, the packets all have different payloads, which suggests a block cipher using IVs, like AES-CBC. The header is consistent: f4 8c 50 05 74 81 8c 85 80 3d 38 58 08 00, so perhaps it can be used to identify what type of encoding?



```
f4 8c 50 05 74 81 8c 85  80 3d 38 58 08 00 45 00   ··P·t···  ·=8X··E·
04 24 3a 66 40 00 40 11  6b 72 0a 2a 00 a1 8c b1   ·$:f@·@·  kr·*····
f9 74 58 25 36 9b 04 10  25 83 f1 d0 04 04 d1 01   ·tX%6···  %·······
00 75 6c 42 39 ce 70 cf  a6 33 93 1d 38 dd 81 00   ·ulB9·p·  ·3··8···
c8 08 48 7c e0 8f 0a 81  eb e9 31 15 83 b9 3d 5b   ··H|····  ··1···=[
cf ed 22 d2 bd 38 37 68  da 1c 6b ae 3d d5 ac 29   ··"··87h  ··k·=··)
6f cf 23 05 d4 d4 db 1f  12 b8 22 e6 56 b7 74 f0   o·#·····  ··"·V·t·
b9 9d 20 c3 ae 27 05 0e  7f 53 f6 56 41 19 ef cc   ·· ··'··  ·S·VA···
bf a7 a9 36 26 f6 6b 6c  4b 01 ef 43 81 53 c0 42   ···6&·kl  K··C·S·B
33 88 77 d0 2f 50 b3 f7  08 dd f6 1e 98 09 f5 4b   3·w·/P··  ·······K
9e b9 b7 0f 0d ce 36 6c  89 a0 f7 c7 8b 30 50 c1   ······6l  ·····0P·
5e ab 7b 72 93 57 fd 73  a7 c6 df 25 a0 dd c8 8b   ^·{r·W·s  ···%····
29 64 f5 ef 31 03 cc e7  e8 d1 94 f4 5e 1e bd 82   )d··1···  ····^···
62 84 bb 24 c8 7f 1b 18  8b 48 52 12 05 be 04 8b   b··$····  ·HR·····
a1 f6 3f 8e 60 b4 a8 b4  23 b3 b7 a5 cc 49 4f 67   ··?·`···  #····IOg
57 0e 94 7d 19 1d e5 11  de 61 19 8c ed 05 80 42   W··}····  ·a·····B
b4 24 a5 49 8d 7f a5 2b  96 19 b9 68 66 cb a7 84   ·$·I···+  ···hf···
d7 7c 46 fa 94 4f 0f 32  ab 2f 9d eb 4c 5f ca 37   ·|F··O·2  ·/··L_·7
b7 3e af 26 6f a0 88 e6  d4 b1 99 55 4a 63 b3 1f   ·>·&o···  ···UJc··
b3 45 f5 f6 ad 19 f3 b0  0c f3 03 93 5f d1 91 3d   ·E······  ····_··=
a4 ac 9a 2a f3 0e 6e 3f  a6 02 5c 69 06 3c 5f 9d   ···*··n?  ··\i·<_·
1f 0f 3c 88 97 2d b2 9b  d8 5f 44 0d 4f ed 6e 6d   ··<··-··  ·_D·O·nm
fd d3 bd 60 15 21 36 02  62 ed 3c 1a 67 bc e9 06   ···`·!6·  b·<·g···
50 94 4d cf 79 bb 2c 07  07 d0 ef cf 53 40 93 ca   P·M·y·,·  ····S@··
c7 2e 86 9f 2f c3 0b 16  75 e3 71 fa 04 66 c3 48   ·.··/···  u·q··f·H
b7 a7 2c 48 b6 27 38 61  b7 4a 40 53 36 43 a1 ce   ··,H·'8a  ·J@S6C··
17 83 15 1b 12 b3 b5 73  1d 44 ff c6 48 1f af e9   ·······s  ·D··H···
6a 08 6a b0 7b b7 e5 bb  3e bd 54 ab 30 5f 7e fc   j·j·{···  >·T·0_~·
3b b8 f4 0b a4 64 af ba  47 19 f1 67 80 e2 3d 36   ;····d··  G··g··=6
de f0 98 f3 e3 e6 8c c2  4c 2e f1 6f 64 f9 e3 67   ········  L.·od··g
```

The main conclusion to be drawn from this traffic analysis is that nothing appears to be sent over plaintext, which was an issue found in a previous CVE. None of the packets sent by the doorbell were in plaintext, and the video stream packets were not in a format where the video was recoverable in wireshark. Focusing on the app login, the app initiates a TCP connection to an amazon ec2 instance, and establishes a TLS connection. Further traffic is encrypted, so it does not appear that the app sends any login messages through plain text.

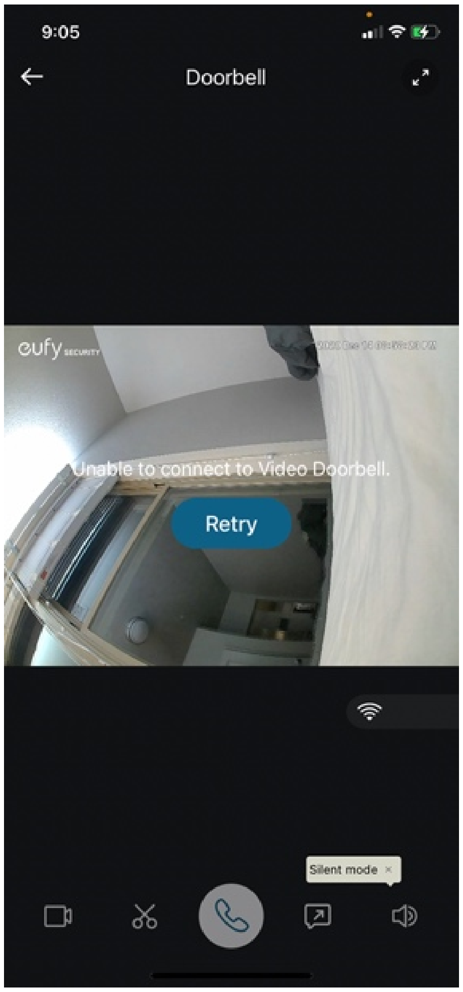


```
263 22.395980395 10.42.0.134   3.18.193.59   TCP      74 47672 → 8789 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=1172329762 TSecr=0 WS=256
264 22.415328724 3.18.193.59   10.42.0.134   TCP      74 8789 → 47672 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 TSval=2488863384 TSecr=1172329762 WS=128
265 22.417989340 10.42.0.134   3.18.193.59   TCP      66 47672 → 8789 [ACK] Seq=1 Ack=1 Win=87808 Len=0 TSval=1172329783 TSecr=2488863384
266 22.421046938 10.42.0.134   3.18.193.59   TLSv1.2 583 Client Hello
267 22.439812577 3.18.193.59   10.42.0.134   TCP      66 8789 → 47672 [ACK] Seq=1 Ack=518 Win=28032 Len=0 TSval=2488863408 TSecr=1172329785
268 22.442523164 3.18.193.59   10.42.0.134   TLSv1.2 1514 Server Hello
269 22.442554613 3.18.193.59   10.42.0.134   TLSv1.2 789 Certificate, Server Key Exchange, Server Hello Done
270 22.444267366 10.42.0.134   3.18.193.59   TCP      66 47672 → 8789 [ACK] Seq=518 Ack=1449 Win=90624 Len=0 TSval=1172329810 TSecr=2488863412
271 22.444330452 10.42.0.134   3.18.193.59   TCP      66 47672 → 8789 [ACK] Seq=518 Ack=2172 Win=93440 Len=0 TSval=1172329810 TSecr=2488863412
272 22.455409962 10.42.0.134   3.18.193.59   TLSv1.2 192 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
273 22.474732767 3.18.193.59   10.42.0.134   TLSv1.2 117 Change Cipher Spec, Encrypted Handshake Message
274 22.476168306 10.42.0.134   3.18.193.59   TLSv1.2 156 Application Data
275 22.497635603 3.18.193.59   10.42.0.134   TLSv1.2  99 Application Data
276 22.497675309 3.18.193.59   10.42.0.134   TLSv1.2  97 Encrypted Alert
277 22.497691240 3.18.193.59   10.42.0.134   TCP      66 8789 → 47672 [FIN, ACK] Seq=2287 Ack=734 Win=28032 Len=0 TSval=2488863467 TSecr=1172329844
278 22.530897568 3.18.193.59   10.42.0.134   TCP      66 [TCP Retransmission] 8789 → 47672 [FIN, ACK] Seq=2287 Ack=734 Win=28032 Len=0 TSval=2488863509 TSecr=1172329844
279 22.600801953 10.42.0.134   3.18.193.59   TCP      78 47672 → 8789 [ACK] Seq=734 Ack=2288 Win=93440 Len=0 TSval=1172329967 TSecr=2488863466 SLE=2287 SRE=2288
280 22.602149093 10.42.0.134   3.18.193.59   TLSv1.2  97 Encrypted Alert
281 22.602060840 10.42.0.134   3.18.193.59   TCP      66 47672 → 8789 [RST, ACK] Seq=765 Ack=2288 Win=93440 Len=0 TSval=1172329968 TSecr=2488863400
```

C. Deauthentication

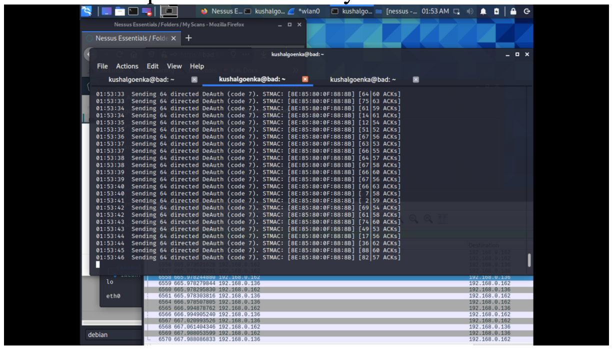
Since the Eufy Doorbell and Homebase system does not use the Weave Daemon, we decided to try a different approach where we sent de-authentication packets to the HomeBase using aircrack-ng. This attack vector does not need an attacker to have access to the internal NAT that the security system is on and is susceptible to an attacker who is in close proximity or one that has a suitable range on their wireless card. Another pre-requisite to conduct such an attack was that the wireless card had to allow monitor mode.

We saw that almost immediately after sending deauth packets, after about 100 such packets, the Eufy Security App on the phone was unable to connect to the Video footage from the camera. Previously, if the camera were to detect a moving object, or a human in its field of view, it sent a notification to the phone notifying the user that there was activity detected, along with a use configurable short length clip of the event. We saw that our deauthentication attack led to no such notifications. Further, when attempting to access the live video on the camera, we saw multiple error messages such as P2P Connection failure and "Unable to connect to video doorbell and asked us to retry.



This connection error continued as long as the deauth packets were being sent to the HomeBase. This been said, after connection was restored, we saw notifications on the app, along with a recording of the entire event. We hypothesize that since the connection between the Homebase and the Doorbell camera is via Low-Power Bluetooth, on detecting activity, the camera is able to send information to the local storage of the base unit which stores the data and notifications and buffers the same until its able to send it to the victim of the attack. With enough deauthentication packets, we believe that we reach a limit to an internal buffer and hence it takes approximately a minute for the video connection to get restored. Although real time notifications are hampered, we think it's a good feature that the HomeBase is able to send notifications after internet connectivity is restored.

An improvement to this implementation could be to send the user notifications if the HomeBase loses internet connectivity. Since the HomeBase unit is powered through a wall outlet, it would be reasonable to send periodic acks to Eufy Servers to ensure that there is connectivity with the security system, and if within a specific timeout there is no such response, send a notification to the user, notifying them of the issue. This would be helpful if the victim is not at home and would need to be notified of such activity, and connectivity issues to keep them informed. Another interesting approach that leads to a similar issues as deauthentication packets was when we Spoofed the local IP address and Mac Address of the Homebase connected to the router. Looking at packet captures on Wireshark we noticed that the our spoofed network interface card was receiving packets from Eufy Servers hoping to request for locally stored video footage. We hypothesize that packets are originally meant for the HomeBase are accidentally sent to our spoofed attacker, and consequently the packets are dropped and the HomeBase is never able to successfully send the data to the user's phone via the Eufy Servers.



```
01:53:33  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [64|60 ACKs]
01:53:33  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [75|63 ACKs]
01:53:34  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [61|59 ACKs]
01:53:34  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [14|61 ACKs]
01:53:35  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [12|54 ACKs]
01:53:35  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [51|52 ACKs]
01:53:36  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [67|56 ACKs]
01:53:37  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [63|53 ACKs]
01:53:37  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [66|55 ACKs]
01:53:38  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [64|57 ACKs]
01:53:38  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [67|58 ACKs]
01:53:39  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [66|60 ACKs]
01:53:39  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [67|56 ACKs]
01:53:40  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [66|63 ACKs]
01:53:40  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [ 7|58 ACKs]
01:53:41  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [ 2|59 ACKs]
01:53:42  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [69|54 ACKs]
01:53:42  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [61|58 ACKs]
01:53:43  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [74|60 ACKs]
01:53:43  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [49|53 ACKs]
01:53:44  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [17|56 ACKs]
01:53:44  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [36|62 ACKs]
01:53:45  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [88|60 ACKs]
01:53:46  Sending 64 directed DeAuth (code 7). STMAC: [8E:85:80:0F:88:8B] [82|57 ACKs]
```

```
6558 665.978244880 192.168.0.162      192.168.0.136
6559 665.978279844 192.168.0.136      192.168.0.162
6560 665.978295830 192.168.0.162      192.168.0.136
6561 665.978303816 192.168.0.136      192.168.0.162
6564 666.978507805 192.168.0.136      192.168.0.162
6565 666.994878762 192.168.0.162      192.168.0.136
6566 666.994905240 192.168.0.136      192.168.0.162
6567 667.020993526 192.168.0.136      192.168.0.162
6568 667.061404346 192.168.0.162      192.168.0.136
6569 667.988053599 192.168.0.162      192.168.0.136
6570 667.988086833 192.168.0.136      192.168.0.162
```

Nessus Essentials / Folders / My Scans - Mozilla Firefox
kushalgoenka@bad: ~
File Actions Edit View Help
lo
eth0
debian

D. Serial Connection Vulnerabilities:

We were able to successfully replicate CVE-2019-3950 where the Arlo Basestation firmware contained a hard coded username and password combination that allowed root access to the device when an onboard serial interface was connected to it. We found that the default password was not randomized and was simply Admin, with the username Admin as well.

Looking at the Homebase more closely, we noticed that it had 4 ports available for Serial interfacing, these included the Ground, Tx, Rx and 3.3v for power.

Since the firmware updates on the Homebase and both our doorbell cameras were automatic and rare. As a result, we had no control over the timeline, making it difficult to capture the traffic to ascertain which servers these devices were getting their firmware from, and if there were any potential security concerns as prior CVE's have indicated. A good method to follow for further research could be to unplug the device until there is a software update available, and continuously capture packets from the device until the devices are updated. This will allow us to further conduct deep packet analysis, and determine information about the firmware such as where its originating from, whether it's encrypted, etc.

To get a deeper look at the firmware, we attempted to reverse engineer the data on the device by accessing it via the serial interface in recovery mode. We were able to obtain three folders/files. The flash mtd0 partition which was about 32 MB in size, and files and scripts in the etc/ and sbin/ directories. Looking at the files in the etc and sbin folders, we didn't find anything interesting in them. However, there did exist a etc/passwd file. This had the below line:

admin:rC0FditfQ9ix2:0:0:Adminstrator:/:/bin/sh

This prompted us to try and crack the hashed password, via john the ripper, and a simple query revealed the password to be admin.

The password, although encrypted, was too easy to crack, and we believe that this is probably hardcoded for all HomeBase units, which could lead to potential security vulnerabilities.

In recovery mode, we noticed that the ethernet cable was still live and had network access, allowing files to be sent via tftp and telnet for further analysis and reverse engineering.

We found the mtd0 flash partition to be the most interesting find via the serial interface, and its large size indicated that it contained most of the information that we were after.

We used binwalk to access the structure of the files stored in this partition and this revealed a JFFS2 partition, along with compressed files, and a couple of linux partitions as well. We attempted to reverse engineer this and extract the contents but have since been unsuccessful at doing so. We're still actively working on the same, and hoping to learn more about the device after extracting the contents in this flash partition.

To gain further insight about the contents, we used binwalk to plot the entropy of the files in the device. Values very close to 1 could indicate that the files are either compressed, or encrypted. Below is a plot of the entropy graph of the mtd0 partition.

These sharp spikes seem to indicate that the contents are mostly compressed files and this was confirmed via manual inspection as well.

E. Comparing with Prior CVEs:

Our primary goal for our project was to determine if there are cross-manufacturer vulnerabilities that exist and if they do, how quickly are these discovered and patched. We first looked at the manufacturer's firmware update records of the Doorbell, the Homebase as well as other products in its security lineup. These update records didn't explicitly mention any security vulnerabilities that had been patched by the update, and were not linked to any known CVE's.

In our experience when attempting to replicate CVE's and determine if a device and associated software had similar vulnerabilities to those from other manufacturers, we noticed that those vulnerabilities were very specific to the device and its infrastructure in question. This made it very difficult to accurately reproduce the same conditions and especially because the Eufy devices didn't use certain protocols as those used by Ring, Nest, etc.

Furthermore, another issue we discovered was that known exposure notices are very brief, and don't necessarily go into the details of what the vulnerability was, and definitely do not provide additional information about how previous researchers went about discovering the vulnerability. In most cases CVEs directed us to broken links to the formal reports published by the researchers, increasing the difficulty. We attempted to reverse engineer the vulnerabilities as much as possible and where we were unable to, we took the general idea of attack and attempted to craft it for our device and manufacturer.

F. Exploring Denial of Service Attacks:

Since the Eufy Doorbell and Homebase system does not use the Weave Daemon, we decided to try a different approach where we sent de-authentication packets to the HomeBase using aircrack-ng. This attack vector does not need an attacker to have access to the internal NAT that the security system is on and is susceptible to an attacker who is in close proximity or one that has a suitable range on their wireless card. Another pre-requisite to conduct such an attack was that the wireless card had to allow monitor mode.

We saw that almost immediately after sending deauth packets, after about 100 such packets, the Eufy Security App on the phone was unable to connect to the Video footage from the camera. Previously, if the camera were to detect a moving object, or a human in its field of view, it sent a notification to the phone notifying the user that there was activity detected, along with a use configurable short length clip of the event. We saw that our deauthentication attack led to no such notifications. Further, when attempting to access the live video on the camera, we saw multiple error messages such as P2P Connection failure and "Unable to connect to video doorbell and asked us to retry.

This connection error continued as long as the deauth packets were being sent to the HomeBase. This been said, after connection was restored, we saw notifications on the app, along with a recording of the entire event. We hypothesize that since the connection between the Homebase and the Doorbell camera is via Low-Power Bluetooth, on detecting activity, the camera is able to send information to the local storage of the base unit which stores the data and notifications and buffers the same until its able to send it to the victim of the attack. With enough deauthentication packets, we believe that we reach a limit to an internal buffer and hence it takes approximately a minute for the video connection to get restored. Although real time notifications are hampered, we think it's a good feature that the HomeBase is able to send notifications after internet connectivity is restored.

An improvement to this implementation could be to send the user notifications if the HomeBase loses internet connectivity. Since the HomeBase unit is powered through a wall outlet, it would be reasonable to send periodic acks to Eufy Servers to ensure that there is connectivity with the security system, and if within a specific timeout there is no such response, send a notification to the user, notifying them of the issue. This would be helpful if the victim is not at home and would need to be notified of such activity, and connectivity issues to keep them informed. Another interesting approach that leads to a similar issues as deauthentication packets was when we Spoofed the local IP address and Mac Address of the Homebase connected to the router. Looking at packet captures on Wireshark we noticed that the our spoofed network interface card was receiving packets from Eufy Servers hoping to request for locally stored video footage. We hypothesize that packets are originally meant for the HomeBase are accidentally sent to our spoofed attacker, and consequently the packets are dropped and the HomeBase is never able to successfully send the data to the user's phone via the Eufy Servers.

G. Serial Connection Vulnerabilities:

We were able to successfully replicate CVE-2019-3950 where the Arlo Basestation firmware contained a hard coded username and password combination that allowed root access to the device when an onboard serial interface was connected to it. We found that the default password was not randomized and was simply Admin, with the username Admin as well.

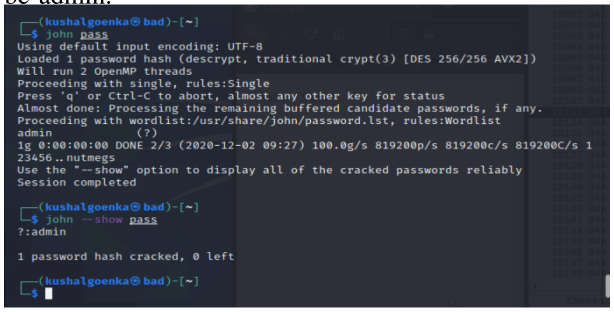
Looking at the Homebase more closely, we noticed that it had 4 ports available for Serial interfacing, these included the Ground, Tx, Rx and 3.3v for power.

Since the firmware updates on the Homebase and both our doorbell cameras were automatic and rare. As a result, we had no control over the timeline, making it difficult to capture the traffic to ascertain which servers these devices were getting their firmware from, and if there were any potential security concerns as prior CVE's have indicated. A good method to follow for further research could be to unplug the device until there is a software update available, and continuously capture packets from the device until the devices are updated. This will allow us to further conduct deep packet analysis, and determine information about the firmware such as where its originating from, whether it's encrypted, etc.

To get a deeper look at the firmware, we attempted to reverse engineer the data on the device by accessing it via the serial interface in recovery mode. We were able to obtain three folders/files. The flash mtd0 partition which was about 32 MB in size, and files and scripts in the etc/ and sbin/ directories. Looking at the files in the etc and sbin folders, we didn't find anything interesting in them. However, there did exist a etc/passwd file. This had the below line:

admin:rC0FditfQ9ix2:0:0:Adminstrator:/:/bin/sh

This prompted us to try and crack the hashed password, via john the ripper, and a simple query revealed the password to be admin.



```
┌──(kushalgoenka㉿bad)-[~]
└─$ john pass
Using default input encoding: UTF-8
Loaded 1 password hash (descrypt, traditional crypt(3) [DES 256/256 AVX2])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
admin            (?)
1g 0:00:00:00 DONE 2/3 (2020-12-02 09:27) 100.0g/s 819200p/s 819200c/s 819200C/s 1
23456..nutmegs
Use the "--show" option to display all of the cracked passwords reliably
Session completed

┌──(kushalgoenka㉿bad)-[~]
└─$ john --show pass
?:admin

1 password hash cracked, 0 left

┌──(kushalgoenka㉿bad)-[~]
└─$ 
```

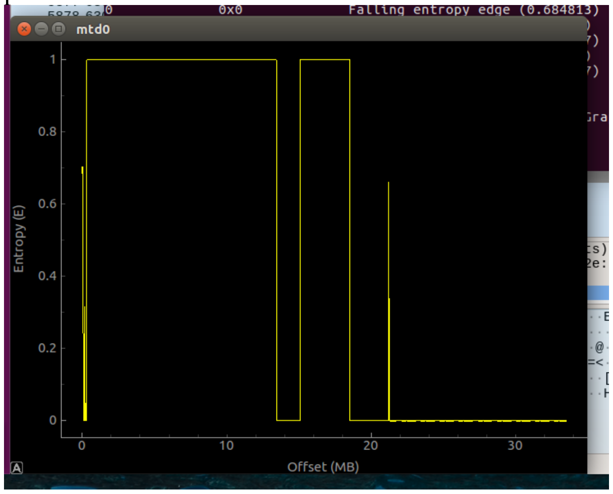
The password, although encrypted, was too easy to crack, and we believe that this is probably hardcoded for all HomeBase units, which could lead to potential security vulnerabilities.

In recovery mode, we noticed that the ethernet cable was still live and had network access, allowing files to be sent via tftp and telnet for further analysis and reverse engineering.

We found the mtd0 flash partition to be the most interesting find via the serial interface, and its large size indicated that it contained most of the information that we were after.

We used binwalk to access the structure of the files stored in this partition and this revealed a JFFS2 partition, along with compressed files, and a couple of linux partitions as well. We attempted to reverse engineer this and extract the contents but have since been unsuccessful at doing so. We're still actively working on the same, and hoping to learn more about the device after extracting the contents in this flash partition.

To gain further insight about the contents, we used binwalk to plot the entropy of the files in the device. Values very close to 1 could indicate that the files are either compressed, or encrypted. Below is a plot of the entropy graph of the mtd0 partition.



These sharp spikes seem to indicate that the contents are mostly compressed files and this was confirmed via manual inspection as well.

H. Man in the Middle Attack:

To make it easier to sniff on network packets, while we did utilize an additional machine as a router which the Doorbell Camera and/or BaseStation were connected to, this was under the assumption that an adversary would be able to successfully compromise the router's integrity. For a more realistic situation, we also tested the feasibility of capturing packets, with the potential to analyze them later, with a Man in the Middle Attack. We conducted this via ARP Poisoning, and placing ourselves between the router and the HomeBase. This allowed us to capture packets similar to how we were able when the HomeBase was connected to a network that we controlled.

We also placed ourselves between the phone on the network and the router, and confirmed our hypothesis about the external servers which the HomeBase sends video information to. We saw the same IP addresses, and hostnames of the EC2 servers as we did with the packets captured during the communication with the HomeBase.

I. Privacy Concerns:

We found interesting analysis to the claims made by Eufy touting the HomeBase+Doorbell Security system as being secure, and privacy safe because the video is stored on your local device and not on external servers. However, our packet capture analysis tells us that the data is actually sent via Eufy owned, Amazon's EC2 servers when a user attempts to access their live camera or stored camera footage. This might defeat the purpose of local storage and could mislead users who were made to believe that there is absolutely no external access to their personal information and videos. This becomes even more concerning when we talk about other camera devices on the same network that are connected to the HomeBase.

J. CVEs which we were not able to reproduce:

CVE-2019-5043, refers to a DOS vulnerability that exists in the Weave daemon of Nest Cameras. Weave is a protocol for setup and initial communications with other Nest Devices over TCP, UDP, Bluetooth and 6lowpan. Since the Eufy Camera setup does not use this specific protocol, we were unable to see if such a vulnerability exists in its cameras.

CVE-2019-9483, where researchers were able to obtain audio and video data.

Amazon Ring Doorbell before 3.4.7 mishandles encryption, which allows attackers to obtain audio and video data, or insert spoofed video that does not correspond to the actual person at the door

VI. Results

VII. Future Work and Improvements

The largest next step would be to conduct these experiments across a much larger range of devices and manufacturers, because our goal is to determine

With more time, expertise, and equipment, we would consider launching active man in the middle attacks to try to replay past packets. A previous CVE shows that sending past video packets would trick the device into thinking someone was at the door. We suspect that this attack would not succeed because we know one of the first connections the doorbell makes is to a time server to get the current time. This implies that the current time is probably included in the video stream packets, so the app and web server would know to discard packets that are too old.

Another angle of attack we were not able to consider this time was the Bluetooth connection between the phone app and doorbell. The vulnerability we would be looking for is an older version of Bluetooth being used that uses weaker encryption standards. It is quite possible that the doorbell supports older versions of Bluetooth to maintain compatibility with older smartphones, so we could potentially launch a downgrade attack.

In the future, we would also try to explore the Android App apk and extract the contents to analyze it. In particular, we would want to run it in a debugger, to try to analyze the cryptographic libraries being called and what video encoding is being used so we have a better chance of deciphering the video stream and determine if they are following safe practices.

Get extra phones without other background apps so when we try to analyze traffic, there isn't a lot of other packets to sort through. Purchase better WiFi sniffers, like wifi pineapple, and setup a computer without background processes that are sending traffic.

REFERENCES

Below are the references for our work above.

REFERENCES

[1] https://yourthings.info/method/

[2] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9483

[3] G. Kambourakis, C. Kolias and A. Stavrou, ”The Mirai botnet and the IoT Zombie Armies,” MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, 2017, pp. 267-272, doi: 10.1109/MILCOM.2017.8170867.

[4] https://www.grandviewresearch.com/industry-analysis/doorbell-camera-market

[5] Cisco: The Internet of Things Reference Model, http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf

[6] Zhang G., Yan c., Ji X., Zhang T., Zhang T., Xu W.DolphinAttack: Inaudible Voice Commands. InProceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security(Dallas,TX, USA, 2017). https://acmccs.github.io/papers/p103-zhangAemb.pdf

[7] Apthorpe N., Reisman D., Sundaresan S., Narayanan A., Feamster N.Spying on the Smart Home:Privacy Attacks and Defenses on Encrypted IoT Traffic. 2017. https://arxiv.org/pdf/1708.05044.pdf

[8] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9483

[9] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-3984

[10] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-3950

[11] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-3949

[12] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5043

[13] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-3988

[14] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-4400

[15] https://www.forbes.com/sites/aarontilley/2017/03/22/this-smart-doorbell-was-accidentally-sending-data-to-china-until-people-started-freaking-out/6461d9075984

[16] https://www.theverge.com/circuitbreaker/2018/5/11/17345972/ ring-smart-doorbell-password-change-revoke-app-permission-access

[17] https://www.bitdefender.com/files/News/CaseStudies/study/294/Bitdefender-WhitePaper-RDoor-CREA3949-en-EN-GenericUse.pdf?clickid=06U0VpxjrxyLWmnwUx0Mo3bwUkExpsw5sz6kyA0irgwc=1MPid=10078cid=aff

[18] https://www.consumerreports.org/video-doorbells/data-security-data-privacy-gaps-found-in-video-doorbells/

[19] B. Marczak, J Scott-Railton https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/

[20] https://shop.hak5.org/products/wifi-pineapple