Title

Stealing, Spying, and Abusing: Consequences of Attacks on Internet of Things Devices

Journal

IEEE Security & Privacy, 17(5)

ISSN

1540-7993

Authors

Valente, Junia
Wynn, Matthew A
Cardenas, Alvaro A

Publication Date

2019

DOI

10.1109/msec.2019.2924167

Peer reviewed

Stealing, Spying, and Abusing: Consequences of Attacks to IoT Devices

Junia Valente, Matthew A. Wynn
The University of Texas at Dallas
{juniavalente, mwynn}@utdallas.edu

Alvaro A. Cardenas
University of California, Santa Cruz
alvaro.cardenas@ucsc.edu

***Abstract*—We study the security practices of a diverse set of IoT devices with different architectures. We find vulnerabilities that can be exploited to launch novel attacks. These real-world implications of IoT attacks show the risks associated with these new technologies and can help us articulate the need for better security practices.**

I. Introduction

It is well-known that most consumer IoT devices are vulnerable, partly because consumers and developers of IoT do not understand the new threats enabled by these devices. We believe that by understanding the potential real-world consequences of attacks to IoT devices, we can better articulate what is at stake when these devices are compromised, and justify calls for improving their security. In this paper we analyze and expand our lessons learned from studying the security and privacy of several IoT devices, and show the unique new capabilities they give the attacker [1]–[4].

In particular, we study drones, smart children toys, Internet-connected cameras, and smart intimate devices. We found several vulnerabilities in these devices, and reported them to respective vendors and CERT/CC, resulting in the following assigned Common Vulnerability & Exposure (CVE) numbers—that were published in the National Vulnerability Database (NVD): CVE-2017-8865, CVE-2017-8866, CVE-2017-8867, CVE-2017-3209, CVE-2015-8286, CVE-2015-8287, CVE-2019-12509, CVE-2017-14486, CVE-2017-14487. These vulnerabilities give the attacker the ability to (1) hijack a drone (stealing it, crashing it, or even harming a person with it), (2) turn on a nearby drone and ask it to stream video and microphone data to spy on a neighbor, (3) take control of a children's toy to insult the child or to have inappropriate conversations with the child, using the *trusted voice* of the toy, (4) take control of the toy to spy on the household, (5) interact with a stolen/or resold toy to extract sensitive information the child once shared with their toy, (6) take control of Internet-connected cameras to spy on a household, (7) identify the existence of sensitive devices (e.g., vibrators) nearby and obtain private information of a person's intimate decisions, and (8) impersonate a trusted partner and allow an attacker to commit remote sexual assault. We also created videos to illustrate and clarify some of these attacks, including the voice attacks on the dino[1], a takeover attack on the drone[2], and attacks on intimate devices[3].

These attacks show how IoT technologies are challenging our cultural assumptions about security and privacy, and will hopefully motivate more emphasis in the security and privacy practices by IoT developers and designers.

II. IoT Devices and Architectures

The devices we analyze in this paper represent a wide variety of IoT interactions and configurations. Some devices act as a Wi-Fi access point so mobile phones or computers can connect and interact directly with them (without Internet connectivity); these include drones, Network Video Recorders (NVR) and smart bulbs. The threat in this operational mode is one of an attacker within Wi-Fi range of the device. This architecture is illustrated in Fig. 1(a).

Consumers can interact with several IoT devices remotely through an app in their mobile phones. For example, the surveillance cameras we study in this paper can be monitored and controlled remotely through an app in the mobile phone or computer of the customer. In this case the essential security question is the authentication of the remote user with the IoT device. This architecture is illustrated in Fig. 1(b).

Some IoT devices can interact directly with the user and the cloud without mobile phone interactions. This category of devices include smart assistants like the Amazon Echo and intelligent children toys like the Hello Barbie or the CogniToys Dino. The threat here are remote attackers in the network who try to spy, or impersonate the cloud to the device. This architecture is illustrated in Fig. 1(c).

A slight variation of the previous architecture is one where the remote user does not interact with the device directly, but the connections are mediated through another mobile device. The intimate devices we study in this paper are an example of this architecture, where the IoT device connects wirelessly—via Bluetooth, Bluetooth Low Energy (BLE), ZigBee, Z-Wave, etc.—to an app in a mobile device. This app then mediates all communications of the IoT device to the outside world. The key security questions here are how mobile apps are authenticating the users of each platform, and what can nearby

[1]https://goo.gl/ApTRPj
[2]https://youtu.be/bAxPBNDeCmM
[3]https://youtu.be/IwMt6dVkRyc and https://youtu.be/B9JmBKQikE0

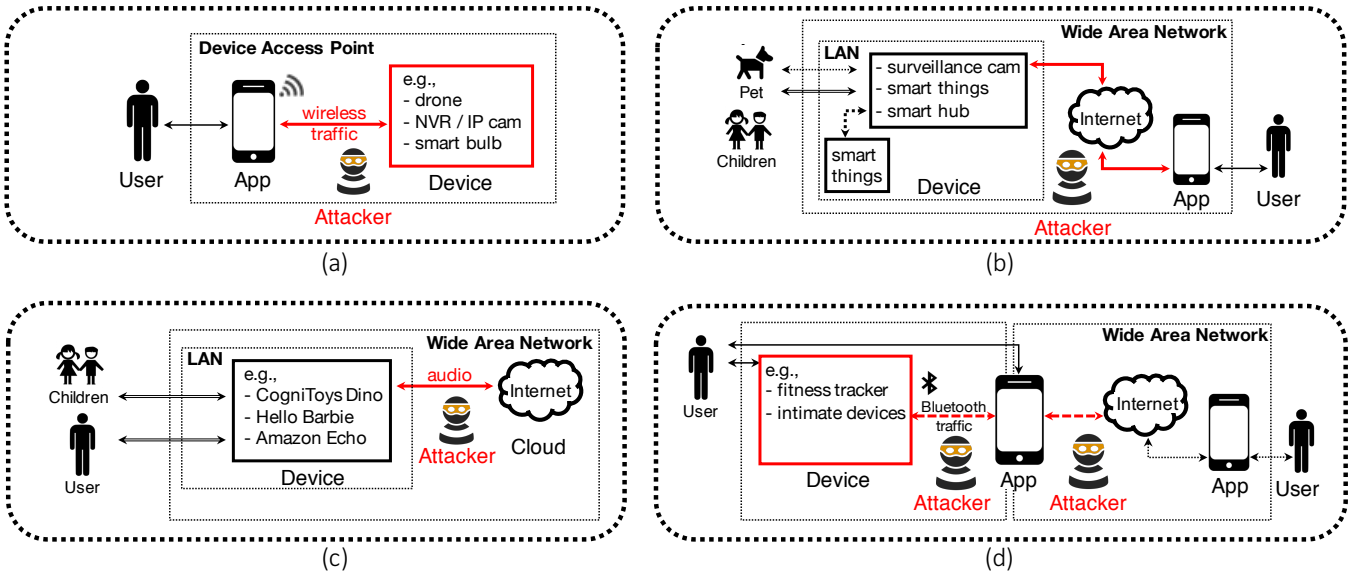


Fig. 1: Summary of IoT architectures

wireless attackers (BLE) infer or attack. This architecture is illustrated in Fig. 1(d).

We now describe attacks in each architecture, their implications, and the lessons learned from our study.

III. Smart Toys for Children

A growing consumer technology is the use of Internet-connected toys for children, as they can help children remain engaged in many types of subjects and improve their problem-solving skills, but new advances and devices can also bring new challenges and problems. Children's brains are in development and are easily molded to develop ideas and norms based on their early experiences. For example, if children are tracked with the help of IoT devices, they will grow up normalizing this type of surveillance. Smart toys that listen to children and interact with them have been growing in popularity; however this rise has also been accompanied by a variety of privacy fears [5], vulnerabilities [6], and government recommendations [7].

We focus our discussion to Internet-connected CogniToys `dino` devices—sold by the U.S. company Elemental Path. This device follows the architecture in Fig. 1(c). While most of the concerns about smart toys have focused on privacy, our audio injection attack can be potentially more dangerous as it is targeting young children who are more vulnerable to deception and who presumably trust the smart toy. In our attack, a criminal can inject audio to the device using the trusted voice of the `dino` to have conversations with the child, learning intimate details of the child or household lifestyle. The attacker can later interact with the child in person appearing as a trusted person by leveraging information gathered with the attack. Other attacks can tell children to open the door to their homes, to drink poisonous elements, or to insult the child and lower their self-esteem.

We discovered our attack by analyzing the encrypted real-time transport protocol (`RTP`) traffic—which transmits audio between the `dino` device and cloud. In particular we noticed 16 repeated patterns in the encrypted traffic, and concluded that each pattern was derived as result of using one of 16 possible symmetric keys to encrypt `VoIP` traffic. To confirm, we noticed that the `dino` sends `VoIP` session initialization packets specifying a key index (ranging from 0 to 15) each time before starting to send encrypted traffic; and these patterns were consistent with the key index used. For example, every time traffic was initialized under key index 0, the repeated pattern was `bbc7-2ed6-b4dd-dbbe-c35b-d9fd-8018-5105`. These patterns are the same for all `dino` devices we have in our lab, implying that all `dinos` shared the same keys with each other.

Instead of trying to break the keys, an easier attack is to simply buy another `dino` device and use the new toy device as a *decrypting tool*, thereby effectively implanting a microphone in the targeted home. To use the attacker's device as a decrypting tool, the device needs to use the *same* symmetric key. The attacker can turn on their device, and keep turning it off and on, until the attacker sees the same pattern in the encrypted traffic between the attacker's `dino` and cloud. When the attacker sees the same pattern, it means the attacker's `dino` is using the same symmetric key that encrypted the network traffic it wants to listen. Now, the attacker can replay the encrypted traffic (from the child's `dino` and the cloud) to its `dino` to appear as if the traffic was coming from the cloud; and the attacker's `dino` can correctly decrypt the `VoIP` traffic, and the attacker can hear the child's voice coming out from the attacker's `dino`.

Perhaps more problematic is the fact that a remote attacker can *inject* audio in a target `dino` device. An attacker can record `VoIP` traffic between its `dino` device and the remote server, and then replay the encrypted `RTP` traffic to a target `dino`.

We summarize the voice injection attack in Fig. 2. This allows an attacker to talk directly with the child, even using the dinosaur's voice (so the child think it is talking to the toy and not to a stranger). If the attacker has a recording of the mother or father of the child, the attacker can then replay it and cause the dinosaur to use the voice of the parent.

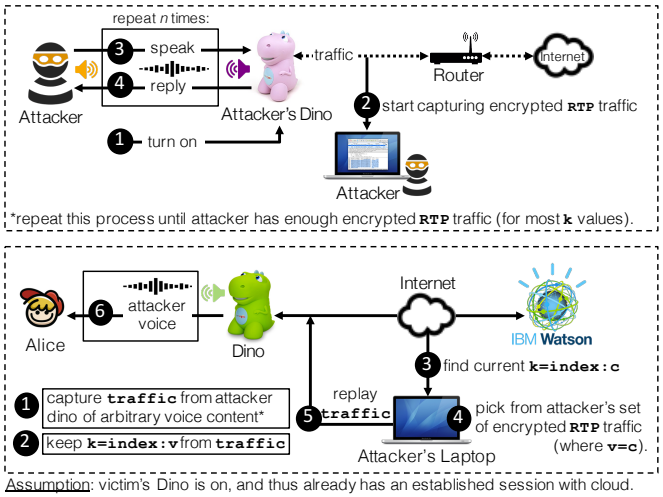


Fig. 2: Voice injection on `dino`: (a) attacker captures encrypted `RTP` traffic while speaking to its own `dino` device; (b) the attacker launches voice injection attack by *replaying* the encrypted `RTP` traffic to a victim's `dino` (after modifying packets' headers). We assume the victim's `dino` is on, and therefore the device has a session established with the cloud.

Another problem with IoT attacks that has received little attention is the problem of stolen or recycled devices. A person in possession of the `dino` device (e.g., if the device is lost, resold, or stolen) can ask the `dino` invasive questions about the previous owner, and the `dino` will reply accurately, revealing privacy-sensitive information. In our experiments, we acquired a `dino` device that belonged to one of the researchers' daughter. We found that we could connect the device to our local area network and then easily connect the device online—without knowing the username and password that the child's parent had used to initially setup the device via the CogniToys app. The issue was that we could use an alternative path: a web server running inside the toy (that was left there for debugging purposes), and this web server allowed us to connect the device online without authenticating the user. After that, we started asking the `dino` questions about the original owner like their name, age, and birthday, and the `dino` happily and helpfully answered all those questions. While we are impressed by the `dino`'s ability to interact intelligently with their users (by far the most "intelligent" of the smart toys we analyzed) these interactions also open the door to privacy invasions.

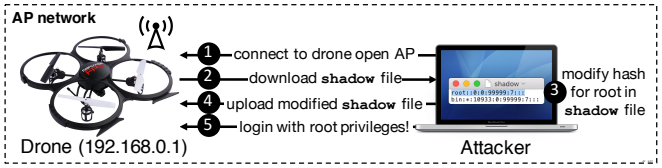
We refer the readers to [1] for additional details.

IV. Consumer Drones

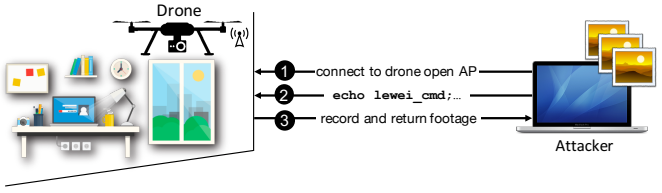
Consumer drones are also changing our notions of privacy and safety. For example, drones have been used to spy on neighbors and children, and drone attacks can also cause safety concerns. While not an attack but an accident, the city of Seattle's first *mishandling a drone in public* charge [8] shows the new type of threats drones can create.

We analyzed the security and privacy threats in drones through a study of the Discovery U818A quadcopters. These drones have a Wi-Fi access point that allows users to control their drone with a mobile app—so these devices follow architecture in Fig. 1(a).

We now describe some of the vulnerabilities that allowed us to attack the drones. Discovery drones allow an *anonymous* `ftp` user to access the drone `ftp` server and modify system files—that only the `root` user should have the privilege to access. We found that a malicious user within Wi-Fi access range of the `drone` can (1) connect to the drone Wi-Fi access point (because it is open), (2) establish a `ftp` connection (since there is no password set by default for the anonymous `ftp` user and it accepts any string as a password), and then (3) see the entire file system and transfer files to/from the drone. Moreover, we found that a malicious user can replace sensitive system files such as `/etc/shadow` to gain further access to the device (i.e., enabling unauthorized users to access a drone mid-air, via the `root` access, and taking it down to intentionally cause accidents). We summarize this attack in Fig. 3a.



(a) An attacker can use the anonymous `ftp` login as a backdoor to gain full `root` access to the device via `telnet`.



(b) Nearby unauthorized users can stalk and spy on drone owners (e.g., take pictures and start video streaming)

Fig. 3: Attacks on Discovery drones

We also found commands that can allow attackers to invade the privacy of drone owners. Discovery drones communicate over commands called `lewei_cmd`. This includes control commands, video streaming commands, a process for transferring files, etc. An attacker can use unknown open ports—namely, undocumented ports `7060` and `8060` to send `lewei_cmd` commands directly to the drone (e.g., take photo, record video, retrieve file names from the SD card mounted

to the drone, transfer files off from the drone to the attacker's machine). During our security analysis, we mapped `lewei_cmd` commands—going through the network—to various functionalities. For example, an unauthenticated user can use a `lewei_cmd` command to request the drone to take photos. This can allow stalkers to turn on spy cameras on their neighbors. We show this attack in Fig. 3b.

We summarize the attacks we successfully tested as follows:

1) **Stealing**: an attacker can launch a de-auth attack to disconnect the phone controlling the drone and start controlling from their phone.
2) **Take-down flying drone**: once attackers have root access to the drone (as explained before) they can send a variety of commands, including powering-off the device and knocking down the drone from the air.
3) **Steal user data**: an attacker can get access to the video feed of the drone, which can affect operators who use drone for commercial purposes.
4) **Spy**: the attacker can create network packets with commands to turn on the drone's camera and return to us footage without the legitimate drone owner noticing this was happening.

We refer the readers to [2] for more details.

V. Intimate Devices

People's intimate lives are also changing given the rise of new IoT devices. Smart intimate devices have been designed to enhance relationships (e.g., military couples separated by distance) and can aid couples seeking medical help; however, the functionalities of these devices open the door to new security, privacy, and safety concerns. These devices can also enable new forms of sexual assault.

For example, intimate devices that offer control over the Internet, may be hijacked by an impostor to control the device while pretending they were the trusted partner. These acts constitute a new form of sexual assault because the victim is participating without affirmative consent (the victim has been misled by the attacker into thinking she is interacting with the trusted partner). For example, under California law, it is a crime when the victim submits under the belief that the person committing the act is someone known to the victim other than the accused, and this belief is induced by any artifice, pretense, or concealment practiced by the accused, with the intent to induce the belief. Similarly, the United States Uniform Code of Military Justice includes in the definition of sexual assault the following forbidden means to achieve non-consensual practices: "inducing a belief by any artifice, pretense, or concealment that the person is another person." Therefore, impersonating a trusted partner through these intimate devices can be a form of sexual assault (e.g., California Penal Code, Part 1, Title 9, Chapter 1, Section 261).

As far as we are aware, we were the first researchers to discuss the potential of remote sexual assault, and showed practical attacks demonstrating this threat on commercially available products (we refer the readers to our initial work [3]).

In addition to *remote* violence concerns, these devices also pose significant privacy concerns. Mobile apps and IoT devices are collecting physical data of diverse human activities; perhaps no other information captured by IoT devices is as sensitive as the data collected by intimate devices. These data can be a top target for stalkers and even more problematically, the data collected by the developer of the app or IoT device might be requested (or hacked) by governments with repressive views that punish as deviant certain private preferences.

We studied two wearable devices: `vibease` and `OhMiBod`. Both devices communicate with an app via Bluetooth, and the app communicates with the manufacturer servers. These devices follow architecture in Fig. 1(d).

Vibease. A trusted partner can remotely control the `vibease` device through an app. However, the vulnerabilities we found (see [3] for more details) give an attacker the ability to *impersonate* a trusted partner and then send specific control requests. The main problem is that the `vibease` Android app (and iOS app) exchanges `xmpp` messages—for sending remote commands and chat communication—with other apps without using encryption. More problematic is that the app was sending authentication tokens to `vibease` servers in the clear by simply encoding the token using `base64`, e.g., `token=base64(username\0password)`. So, when an attacker sees the following plaintext message:

```
<auth xmlns="urn:ietf;params:xml:ns:xmpp-sasl" mechanism="
    PLAIN">AHRlbW9jMgBqNUJYeDk2RWlvM0hEcDRxOFJBZQ==
</auth>
```

the attacker can then decode the plaintext message to retrieve: `temoc2j5BXx96Eio3HDp4q8RAe`. This reveals that the username is `temoc2` and the password is `j5BXx96Eio3HDp4q8RAe`. With this information, unauthorized users can login to any standard `xmpp` client (with the credentials found above) using the domain `chat.vibease.com`. Therefore, besides a remote attacker being able to obtain user credentials, intercept chat messages, and gather sensitive information by sniffing the network traffic, the attacker can then use the user credentials to impersonate the user—by sending and receiving chat messages from a trusted contact, and sending unsolicited commands.

OhMiBod. Similar to the `vibease` device, an attacker can also impersonate a remote user and send commands to `OhMiBod` devices. The vulnerability was with the `OhMiBod` API server—which we found to be disclosing authentication tokens for any `OhMiBod` users we wished to find. The network traffic between the `OhMiBod` app (for both Android and iOS) happens via `TLS`. So, we used `SSLsplit` as a proxy to inspect the TLS traffic (we refer the readers to [3] for implementation details). This revealed API calls for various functionalities: editing user profile, searching for users, requesting control, and chatting with another trusted user. We found vulnerabilities in the search mechanism: when we search for users, the `OhMiBod` API returned the `user_id`, `username`, a few other fields, and more alarming—a `token` field—as we show below:

```
{ "result": {
    "data": {
        "users": [{
```

```
            "country": "United States",
            "email": "2b241506ec773b93276f4c2
                    18346527668991fcfedaa3214496
                    db6a04c257c5dcadd7d2a9564a17
                    8d1a8a20d3615d579789a357ac5a
                    595596efbe7ec8a20e911",
            "gender": "f",
            "state": "AL",
            "token": "$2a$12$b9dDipB8QP1ancyToKJo
                JuywpXAfQ19WOlHBT1BJskv98.K89qV/y",
            "user_id": 61852,
            "username": "temoctest",
            ...[redacted]
            }]
        }
    }
}
```


It became clear to us that these fields identified users between session. Since we can find this information by simply searching for a user, we can then impersonate *any* user. Rather than reverse engineering **OhMiBod**'s websocket protocol (to setup a client that could speak with it), it was easy enough to test attacks by editing the app setting: e.g., an attacker can edit **/data/data/com.ohmibod.remote2/shared_prefs/OMB.xml** (from a rooted Android phone). This file contains information about the currently logged-in user and their **OhMiBod** contacts. After replacing all instances of the logged-in user's **user_id**, **username**, and **token** (with those of a target user), now when the attacker starts the app from their phone, the attacker can request to connect with anyone previously selected as a trusted partner. In summary, an attacker with a rooted Android phone could query the social network database to get information required to impersonate a given victim. As a result, a victim contact sees the request as coming from a presumably trusted partner (but in reality it is coming from a compromised account!), and as such, any consequent chats and interactions will appear as coming from a legitimate user which the victim contact trusts.

In addition to impersonation problems, these devices have new privacy concerns. For example, the fact that someone is carrying or even wearing one of these devices can be done by scanning nearby BLE devices. During our tests, a neighbor's Vizio TV was sending scan requests and the device was more than happy to tell the TV that it was a **vibease** device; even if the device was not in pairing mode. This is a big privacy risk, e.g., if there are only two people in a room and one finds the existence of the device, it will compromise the privacy of the device owner. BLE also provides devices with the ability to hide their real address and to randomize their address to prevent them from being tracked across multiple locations, but the **vibease** does not use these functionalities. A more secure option would be to use **ADV_DIRECT_IND** when not in pairing mode, and to enable the use of random BLE addresses.

VI. Surveillance Cameras

Internet-connected cameras have seen a growing popularity, allowing their users to care for their children, pets, old age relatives, and the security of their homes; and at the same time, it has given corporations a reliable surveillance system for their premises. The security of surveillance cameras however has come into question by multiple reports of unauthorized camera access by attackers, video feeds of multiple cameras openly available online, and the considerable number of compromised cameras making part of IoT botnets (e.g., Mirai [9]). More troubling, similar to insecure children toys, hacked smart cameras can also be used to abuse children and cause psychological distress. In a recent instance, an almost-3-year old would repeatedly tell her mother "there's a monster in my room," and while the mother initially thought the child was having nightmares, one day the mother walked into the room to find their Nest camera (she used as a baby monitor) playing pornography in her daughter's room [10].

In our analysis, we studied the security of the Swann NVW-470 surveillance system, and the LeFun baby monitor. These devices follow architectures in Fig. 1(a) and Fig. 1(b) (depending on how the user chooses to use the cameras).

Swann Surveillance System. The Swann NVW-470 surveillance system comes with two devices: an IP camera and a Network Video Recorder (**nvr**). The Swann **nvr** is a small-sized monitor (similar to a tablet device) for the user to view real-time footage directly from Swann cameras. The **nvr** creates an access point, and the camera automatically connects to this access point. If the user wants to connect these devices to the Internet, then they can switch the **nvr** into *station* mode, and connect both devices to a LAN network. We talk about vulnerabilities we found in the **nvr** and Swann camera next:

(1) Swann NVR. We found our first set of vulnerabilities in cameras by analyzing the firmware of the **nvr**: we found hard-coded passwords for the **root** user in the firmware, allowing us to gain **telnet** access to the surveillance system. Unlike most of the other IoT devices we analyzed, the firmware for the **nvr** was available online in the vendor's website. So, we downloaded a copy and analyzed it. We used the **binwalk** utility to analyze the firmware image. We found that the **nvr** device uses the **cramfs** file system—a compressed read-only file system often used in embedded devices due to its simplicity to save disk space. It was possible to extract and unpack the entire **cramfs** file system because the firmware was not encrypted. At that point, we found that the **nvr** contains a hard-coded password that once extracted can be used to remote login to the device via **telnet** on port **23**. This is possible because the **nvr** devices use the same weak password hard-coded in the firmware copy available online; and can be easily cracked (using tools such as John the Ripper) when an attacker extracts the **/etc/passwd** hash file directly from the firmware. Once the root password is retrieved, it is possible to remote access the device via **telnet**. We succesfully tested the following attacks: e.g., expose video feed from the camera to the Internet, use Busybox utilities (installed on the **nvr**) like **netcat** to create reverse proxies, and further open public-facing ports on someone's router.

(2) Swann IP camera. The Swann camera uses the application-level real-time streaming protocol (**rtsp**) to stream video on ports **554** or **6001**. We found that the **rtsp** service running on the camera does not implement any authentication mechanism. An attacker can take advantage of this lack of authentication to view live feed directly via port **554** or **6001**.

The media stream is uniquely identified and accessed in those ports by a stream URL. All the attacker has to do is to figure out this stream URL. This can be done, for instance, by reverse engineering the `nvr` and looking for hard-coded URLs the `nvr` accesses on the camera. (We also found ways to send well-crafted `xml` packets to the camera, and get the URL as a response). Then, anyone can use the URL to open a `rtsp` stream from applications such as `vlc` or `openRTSP`. The security design flaw here is that authentication is enforced *only* via the normal flow: i.e., when the live feed is accessed through proprietary applications available with the surveillance system, and not when it is accessed directly via the `rtsp` service running on ports `554` and `6001`.

Another prevalent security problem in IoT devices is the `UPnP` protocol. An attacker inside the `nvr` can use the `UPnP` service to expose internal network services to the Internet. In our experiments, we successfully used the `root` access in the `nvr` with the `UPnP` service to not only allow unauthorized users from within the same network as the `nvr` to access the live feed of the camera (via ports `554` or `6001`) but also anyone from the Internet.

LeFun baby monitor. Finally, we found that the mobile app (`MIPC` Android app) for the LeFun camera exposes valid session tokens when communicating to the cloud server under certain scenarios (e.g., to configure the camera, setup cloud storage, associate a new camera to a user account). This enables an attacker to easily impersonate a victim. We noticed two main flaws in this system: the app sends valid session tokens (1) over an insecure channel (via `HTTP`) under some scenarios, and (2) via `GET` methods. Best practices teach us otherwise [11]. Further, the `HTTP GET` requests expose the camera ID which identifies a LeFun camera online. Once a remote attacker hijacks the session token and camera ID, the attacker can *impersonate* the app, and perform any action to appear as if it was the owner of the camera sending commands or viewing the feed from an authenticated user in the app. We illustrate this attack in Fig. 4. The attacker can also use the hijacked session to send commands to cameras: ranging from tilting the camera and controlling camera movements (e.g., to move camera away from an area of surveillance), to changing basic settings (e.g., to change the brightness, to "blind" cameras from capturing the actual scenery). We further noticed that the session tokens did not appropriately expire, and on the contrary, multiple session tokens remained valid simultaneously, giving enough time for an attacker to operate under a hijacked session token.

We refer the readers to [4] for implementation details.

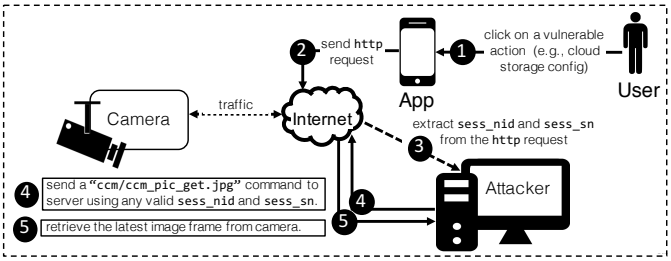


Fig. 4: Attack on LeFun cameras: a remote attacker can eavesdrop on a valid session token (e.g., `sess_sn`) and camera ID (e.g., `sess_nid`). (Because some actions the user takes are transmitted over `http GET` requests). The attacker can use this information to craft valid `GET` requests and capture image frames from an online remote camera.

VII. Findings and Lessons Learned

Studying diverse IoT devices simultaneously allowed us to identify some unique trends and problems. We summarize our results in Table I.

We can see that although most devices implemented encryption, we were still able to get privacy-sensitive information from most of them. We bypassed encryption protections via misconfigured network services, lack of proper key management, or by stealing authentication tokens.

Lack of proper user authentication is also one of the most pervasive problems in the devices we found. Several devices like OhMiBod and LeFun camera had authentication tokens that were sent in the clear at some point (LeFun cameras sent it with certain operations, and the OhMiBod device sent it by mistake in their search API).

Proper user authentication is also essential because some IoT devices will be resold, stolen, or lost. For example for the `dino` children toy, anyone with physical access could connect the device to the Internet (using the troubleshooting available on the vendor's website) and start interacting with the `dino` device to ask personal questions (e.g., when is my birthday? where do I live?), and the `dino` will happily respond to these. The device does not authenticate the user's voice nor the user; so, anyone with physical access can extract sensitive information about the legitimate owner. Another IoT device we analyzed was the Petcube which is an interactive camera device for people to interact with their pets through audio and laser pointers. We found that anyone with physical access to the device could extract the Wi-Fi SSID and password for networks this device was configured to connect to. What we had to do was to turn on the device into the "manual firmware update mode" while connecting the device to our machine via USB. Once the device booted, it appeared as an external hard drive and we could see configuration files containing the list of network credentials. (During a manual firmware update, the user would copy the firmware binary to this external drive).

Another interesting observation was the diversity of how to provision Wi-Fi credentials to IoT devices. Most devices get the Wi-Fi password through a mobile app; however, the connection between the app and the device is not always protected. For example, the app of the Dino connected first to the Dino via a temporary unprotected Wi-Fi network hosted at the IoT device, and the password was sent via an `http` request to a web server in the Dino, therefore an attacker within Wi-Fi range during the provisioning step can capture the credentials. A more secure Wi-Fi provision is done by another smart toy

TABLE I: Summary of our findings

Device	CVEs we disclosed	Lack of user auth (on Device)	Lack of user auth (on App)	Session token disclosure	Lack of encryption	Flaws in the encryption	Misconfigured services	Example of attacks we successfully tested:
CogniToys Dino	CVE-2017-8865 CVE-2017-8866 CVE-2017-8867	✓				✓	✓ (web server on toy)	- Physical attack (to extract sensitive user data from the device) - Eavesdropping attack (to listen what a child speaks to the toy) - Voice injection attack (to manipulate what the device speaks to child)
DBPOWER drone Force1 RC drone USA Toyz drone UDIRC drone	CVE-2017-3209	✓	✓		✓		✓ (`ftp`, `telnet`, other ports on drone)	- Fly-away attack - Replace system files and gain remote access - Control and take down flying drone - Steal user data or spy on owner
Swann NVR	CVE-2015-8286						✓ (`telnet` on NVR)	- Remote root access (by extracting root password from firmware online)
Swann IP camera	CVE-2015-8287	✓					✓ (web server & `rtsp` service on camera)	- Use alternative path to bypass authentication and view live feed
LeFun camera	CVE-2019-12509			✓		✓	✓ (misc.)	- Impersonation attack (to view video feed remotely and change settings)
Vibease	CVE-2017-14486			✓	✓		✓ (`xmpp` service)	- Impersonation attack (in chat sessions or to send commands to device) - Eavesdropping attack (to read chat communication)
OhMiBod	CVE-2017-14487			✓			✓ (cloud server API)	- Impersonation attack

we studied (Hello Barbie) where the web server in the smart toy uses **`https`** instead of **`http`** to send the Wi-Fi password. We also saw mobile devices sending Wi-Fi credentials via Bluetooth. A creative way to load the Wi-Fi password in an IoT device is done by the LeFun camera: they transmit the credentials over sound or images, e.g., the app generates a QR code containing the Wi-Fi password, and this is then read by the LeFun camera.

We also observed that for some devices (e.g., Discovery drones and LeFun camera) there were multiple vendors that sold the same device under different brand names and reused the same mobile applications under different names. For instance, we found 17 apps in the Google Play Store that can also fly and view video streams from the Discovery drones we analyzed. For the LeFun camera, we saw a similar trend: we could use the same credentials—for the LeFun app (**`MIPC`**)—to login to half-a-dozen other Android apps—for other IP cameras—and view the live stream from them. (It seems one camera has multiple vendors and domains that connect to a single cloud system supported by a single company). This creates a vulnerability: a single bug in one of these multiple apps can be used to compromise the system. For example, at the time of this writing the **`MIPC`** app was last updated on Apr/19, meanwhile, another of these apps such as **`Vimtag`** had not been updated since Jan/19, and **`Myannke`** app had not been updated since Sept/18. The reason for these trends is that we have one company designing the core-functionality for the IoT device and then multiple vendors re-purposing the devices under different brand names.

Responsible disclosure. We reached out to all the vendors of affected devices before our publications. Of all the companies, we were surprised that the companies selling intimate devices were the most responsive, and the only ones to patch their devices before the publication of our results. Vibease interacted with us through email, and ultimately added the name of the author who discovered the Vibease vulnerability to their security website as an acknowledgment for helping them make their products more secure[4]. OhMiBod proactively requested conference calls with us so that we could explain our attacks and how to mitigate them. They also patched their device after our discussions.

The company developing the Dinos was also responsive and asked us questions about how to protect the device. We received a couple of emails after the publication of our results stating that they were still working on a patch, but ultimately, we did not notice a security update.

The Swann company focusing on surveillance systems was also responsive and put out a new firmware release claiming to fix the vulnerabilities, however, they stopped replying once we informed them that the vulnerabilities were still present.

We also submitted vulnerability reports to the manufacturers of the drones but we never received a reply. (The only response we received was a thank you message for using their products). Part of the problem with reporting problems to the drone and LeFun camera is that as we noted earlier in the paper, there are several companies (companies A,B,C) that apparently buy barebone kits from a specific manufacturer and then release their own product to the market based on the design of another company (company D) so while we reported the vulnerabilities to companies A, and B, they ultimately may not have the ability to patch the devices, and would have to rely on company D to patch it.

In summary, we reached out to the following companies and received the following replies:

- Toy: Elemental Path—USA. Responded to our emails, asked us how to improve their device, and promised to

[4] https://www.vibease.com/security

work on a patch, but we never saw one.
- Drone: DBPOWER—UK. No response.
- Drone: Force1 RC—USA. No response.
- Drone: USA Toyz—USA. No response.
- Drone: UDIRC Technology—China. No response.
- Camera: Swann—Australia. Responded and released a patch, but our reported vulnerabilities were still present.
- Camera: LeFun Smart Innovation—China. Responded to our emails, and were working on a fix.
- Vibrator: OhMiBod—USA. Successfully patched device.
- Vibrator: Vibease—USA. Successfully patched device and acknowledged our help on their website.

VIII. Conclusions

In this paper we have discussed a set of novel attacks on IoT devices. The variety of devices also shows how their vulnerabilities might be different. For example, since the drones we studied are not meant to talk over the Internet, vendors presumably thought that they did not need to focus on hardening the security of these devices; however, their Wi-Fi access points will give nearby attackers full control to take over the drone and use it to spy or cause physical harm.

On the other hand, since the dinosaur is connected through the Internet, it uses end-to-end encryption (and authentication) between the device and the cloud. This problem can be easily solved with public-key cryptography, but the vendor decided to implement symmetric crypto primitives (presumably for energy efficiency as this device is battery-operated, as many other IoT devices). With symmetric cryptography it is important to keep the secret keys protected, but their implementation allowed an attacker to purchase another device to decrypt Internet-transversing packets, and more worryingly, their algorithm let attackers authenticate as the cloud to the dinosaur, so the attacker could take control of the device, insulting the child or asking the child sensitive questions.

Similarly, the smart vibrators also have different problems. The first device had an authentication token that was transmitted in cleartext during the connection establishment, the second device had a database that returned more information than necessary when discovering friends; while that information would not be available to a regular user of the app, an attacker inspecting the traffic back and forth between the device and the cloud could get this extra information and impersonate the trusted partner of the victim.

Finally, our analysis of the cameras showed a distinct set of vulnerabilities: we were able to find and download the firmware of the Swann devices, and by inspecting the firmware, we found the `shadow` file that we cracked with a regular password cracker to obtain default usernames and passwords (e.g., for the `root` user) for remote connections. Our analysis of cameras also showed another popular problem with IoT devices: `UPnP`. `UPnP` helps the attacker expose internal services on the Internet. `UPnP` is what attackers with partial control of a device use to then expose more services of the device on the Internet and then attack those services.

Our overall results show the importance on protecting IoT devices. For example, while previous work on children toys and intimate devices focused on the privacy problems of these devices, in our work we have shown that in addition to privacy, an attacker impersonating the toy or the trusted partner can cause new consequences previously not considered by developers of these devices. Another interesting observation based on our work with multiple vendors was that the companies selling intimate devices were far more responsive than the other companies we contacted.

We hope our work can help discussions on the importance of securing IoT platforms and in particular, help convince skeptics that some of the risks in consumer IoT devices are significant and can cause previously unanticipated attacks.

Acknowledgments

This research has been partially supported by NSF CNS-1931573 and by a gift from Intel Corporation.

References

[1] J. Valente and A. Cardenas, "Security & privacy in smart toys," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. Dallas, Texas, USA: ACM, 2017, pp. 19–24.

[2] J. Valente and A. Cardenas, "Understanding security threats in consumer drones through the lens of the Discovery quadcopter family," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. Dallas, Texas, USA: ACM, 2017, pp. 31–36.

[3] M. Wynn, K. Tillotson, R. Kao, A. Calderon, A. Murillo, J. Camargo, R. Mantilla, B. Rangel, A. A. Cardenas, and S. Rueda, "Sexual intimacy in the age of smart devices: Are we practicing safe IoT?" in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. Dallas, Texas, USA: ACM, 2017, pp. 25–30.

[4] J. Valente, K. Koneru, and A. Cardenas, "Privacy and security in Internet-connected cameras," in *Proceedings of the International Congress on Internet of Things (ICIOT'19)*. Milan, Italy: IEEE, 2019, pp. 1–8.

[5] 7News, "From Barbies to bears, interactive 'smart toys' could be a hacker's plaything," https://au.news.yahoo.com/a/36421102/from-barbies-to-bears/, 2017.

[6] L. Cooper, "Millions Of Private Messages Between Parents And Kids Hacked In Cloud Pets Security Breach," https://www.huffingtonpost.com.au/2017/02/28/millions-of-private-messages-between-parents-and-kids-hacked-in, 2017.

[7] A. Newcomb, "FBI Warns Parents of Privacy Risks With Internet Connected Toys," https://www.nbcnews.com/tech/security/fbi-warns-parents-privacy-risks-internet-connected-toys-n784126, 2017.

[8] City of Seattle, "City attorney's office prevails in drone case," https://news.seattle.gov/2017/01/13/city-attorneys-office-prevails, Jan. 2017.

[9] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[10] R. Albergotti, "How Nest, designed to keep intruders out of people's homes, effectively allowed hackers to get in," https://www.washingtonpost.com/technology/2019/04/23/how-nest-designed-keep-intruders-out-peoples-homes-effectively-allowed-hackers-get, Apr. 2019.

[11] OWASP Testing Project, "Testing for Exposed Session Variables (OTG-SESS-004)," 2014.