Security & Privacy in Smart Toys

Junia Valente, Alvaro A. Cardenas
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas
{juniavalente, alvaro.cardenas}@utdallas.edu

Abstract

We analyze the security practices of three smart toys that communicate with children through voice commands. We show the general communication architecture, and some general security and privacy practices by each of the devices. Then we focus on the analysis of one particular toy, and show how attackers can decrypt communications to and from a target device, and perhaps more worryingly, the attackers can also inject audio into the toy so the children listens to any arbitrary audio file the attacker sends to the toy. This last attack raises new safety concerns that manufacturers of smart toys should prevent.

1 Introduction

Technology is taking an ever increasing role in children's development. It 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. 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 [11, 17, 26, 29], vulnerabilities [9, 14, 16, 19, 22, 24], research studies [20, 21], and governmental recommendations [12, 28, 31].

To identify the best security and privacy recommendations for smart toys, it is important to understand how these devices can interact with children, and how they can fail; and in particular, it is important to see if these devices can create new unanticipated vectors of attack. To address some of these issues in this paper we study three smart toys, summarize their technologies, and identify new attacks where an adversary can interact directly with children, creating not only privacy problems, but also safety and security problems.

We study the CogniToys Dino [15], Hello Barbie [5], and the Toymail Talkie [7]. All of these devices enable a child to communicate with the toy via voice commands; however they all implement fairly different technologies. These three devices show different approaches for interacting with children and allow us to explore the new safety and privacy threats smart toys may be subject to. The CogniToys Dino has been particularly recognized: it won the grand prize in the 2014 Watson Mobile Developer Challenge which earned a partnership with IBM to develop the technology further [23] and won the 2015 Best use of AI in Education award. Also, it was named as one of the Best Inventions of 2015 by TIME Magazine and a finalist for the prestigious 2017 Toy of the Year Award by the Toy Industry Association [15]. Because we found a vulnerability in how CogniToys Dino was authenticating and encrypting communications between the toy and the cloud, we will focus on this device to illustrate new attack patterns as other devices may be vulnerable to similar attacks in the future.

In particular, we point out that while most of the concerns about smart toys have focused on privacy, our audio injection attack can potentially be more dangerous as it is targeting young children who are more vulnerable to deception and who presumably trust the smart toy. For example, the attacker can inject audio to the device so the smart toy tells children to open the door to their houses, or to change combination locks, or lies about their parents. The attacker can even be mean to the child, insulting their appearance or intelligence and therefore eroding from an early age their self-esteem and their trust of the toy and technology. We uploaded videos illustrating the eavesdropping and voice injection attacks we describe in this paper to a playlist on YouTube[1].

2 Overview of Analyzed Smart Toys

Our three toys (`dino`, `barbie`, and `talkie`) all share two things in common: these devices are equipped with a microphone and embedded speakers, and the user interacts with the device by pressing a physical button and speaking something to the device.

Our security analysis include two main parts: analyzing open ports on the toy devices; and analyzing the network communication between the IoT device and the cloud.

First, unlike other IoT devices (e.g., surveillance systems, consumer drones), we found that these devices do not have ports such as `ftp`, `telnet`, or `ssh` open, and as such they are not vulnerable to the same security weaknesses that may be associated to these services. Instead, they tend to have a web server running over ports `80` or `443` and which are strictly used for Wi-Fi provisioning and configuration of the device. Typically, once the devices are configured and are given the credentials of the local Wi-Fi network, these ports and any open Wi-Fi access points in the toys are closed (unless the user resets the device to factory settings). Further, once these devices are provisioned to join a Wi-Fi network, they communicate with the cloud in two different ways: either by establishing and keeping a connection open with the cloud for the duration in which the device is turned on (even when the device is not in use), or by establishing a new connection each time the device wants to transfer data to the cloud. For the CogniToys `dino` and Hello `barbie` devices, a new connection is established with the server each time the device restarts; while for the Toymail `talkie` device, only each time the device pushes a new voice message (i.e., transfers an audio file) to the cloud or pulls the server for new messages. Then, the connection is immediately closed and the device enters into a sleep mode after the toy finishes sending or receiving audio files (we noticed that the transmission rate for transferring these audio messages is very slow—e.g., may take 5 to 10 seconds depending on the size of the audio message). As a result, the `talkie` device may not receive a new voicemail immediately when a friend or parent sends them until either the device pulls the server again

IoTS&P'17, November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-5396-0/17/11…$15.00
https://doi.org/10.1145/3139937.3139947

[1] https://goo.gl/ApTRPj

(which it periodically does) or the user pushes a physical button on the toy to check for new audio messages.

Secondly, to analyze the network traffic between each device, we created a Wi-Fi hotspot on our computer and provisioned each device to connect to this Wi-Fi hotspot. As such, our computer was placed between the device and the cloud (to act like a man-in-the-middle); and all the network traffic was routed through our machine. Fig. 1 shows an example for this setup when analyzing the encrypted traffic going back and forth between the smart toy and the cloud. Our machine communicates to the cloud by being connected to a router (via Ethernet) with Internet access.

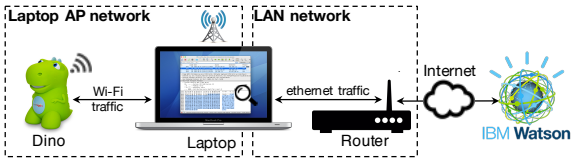


Figure 1. We setup an access point to route all the communication to analyze traffic between smart toys and the cloud.

2.1 How Toys Join Secured Wi-Fi Networks

Because most IoT devices do not have displays nor keyboards, one common way to provision them to join a secured Wi-Fi network involves the user connecting their mobile device to the Wi-Fi access point in the IoT device, and then using an app to send the network information to the web server running in the IoT device (this is done by a variety of devices including the Amazon Echo Dot, smart plugs, smart light bulbs, etc.).

For the `dino`, the provisioning step happens via a proprietary app, or by sending a `http` request directly to the toy's web server via our laptop. Note that the access point is `open` and does not provide encryption; so if a near-by neighbor is *listening* to the traffic during the provisioning, they *will* capture the network password sent in clear text to the device (as shown in the `http` request in Listing 1).

Listing 1. Provisioning `dino`—network info is passed in plain-text:

```
curl -v -H "Content-Type: application/x-www-form-urlencoded" -d "
      __SL_P_S.R=main.html&__SL_P_P.A=SSID&__SL_P_P.B=3&__SL_P_P.
      C=PASSWORD&__SL_P_P.D=2" http://<ip>/profiles_add.html
```

Based on the web server found on the `dino` (e.g., configuration panel) and the toy's MAC address, we conclude that the device is powered by the Texas Instruments (TI) SimpleLink CC3x family of microcontrollers [30] (which is a Wi-Fi Internet-on-a-chip for the deployment of IoT devices). Furthermore, we notice that the device relies on sample applications [10] and follows best practices from a TI white paper [27]. This TI white paper acknowledges that an eavesdropper can intercept the network password during Wi-Fi provisioning, but they argue that the security risks are small because the provisioning happens only once and the attacker might not know when it is happening. However, we argue that given the proliferation of IoT devices using this functionality, the motivation for an attacker to deploy devices listening for network passwords being leaked over plain-text will increase in the next few years.

We found examples in other smart toys that attempt to transmit the Wi-Fi password between the phone and the toy in a `TLS`-protected channel. The `barbie` device runs a web server with `https`, and after a secure connection is established between the `barbie` and the app in the phone, the network SSID and password are sent via the port 443 on the device. However, it has been shown [25] that the secure communication provided by the device could be vulnerable to a connection flaw: an attacker can trick the device to connect to any insecure Wi-Fi network with the name "Barbie" (the attacker can create this Wi-Fi network) and it is then possible to retrieve the network credentials from the proprietary app.

Finally, the `talkie` uses a completely different approach (one we have not found in the dozens of IoT devices we have analyzed before): during the Wi-Fi provisioning step, the `talkie` device uses physical sensors in the device to receive network information entered via the toy mobile app. The information is transmitted to the toy using *visual flashing lights* that the optical sensor in the toy can perceive and further retrieve the credential information embedded in the sequence of flashing lights. In particular, `talkie` uses the BlinkUp technology by Electric Imp [1]: BlinkUp transmits information by rapidly flashing light pulses on th mobile device's screen running a proprietary app, and the data is read by the optical sensor tied to the IoT device's integrated impModule hardware [3].

By not having an access point nor a web server running when it is first boot up, the `talkie` device decreases its attack surface because it prevents near-by devices from connecting and potentially exploiting web vulnerabilities in the toy.

2.2 Communication mechanisms

2.2.1 Deployment Architecture

We observe that IoT-based children toys can be classified in two, (1) *smart toys*, where a child directly interacts with the device (e.g., `dino` which intelligently replies to the child, or `barbie` that responds using pre-recorded phrases), and (2) *connected toys*, where a child uses the device as a means to communicate with their parents or friends. These devices can communicate to the cloud via two architectures: (1) **IoT to cloud**, or (2) **IoT to App (via cloud).**

Architecture 1: IoT to cloud. Devices in this category are configured to connect to a local area network with Internet access. The device must have connectivity to the Internet because typically all the computation (e.g., natural language processing, speech recognition) is performed by a back-end server in the cloud.

As shown in Fig. 2a, a unique characteristic of devices deployed using this architectural model is that the user does not necessarily interact with the device by means of a proprietary app. Instead, the user *physically* interacts with the device, normally by pressing physical buttons on the device, and/or by speaking voice commands to the device. For instance, to communicate with CogniToys `dino` or Hello `barbie`, a child must first press a physical button on the device before speaking to the device.

Architecture 2: IoT to App (via cloud). This IoT architectural model provides two new capabilities: (1) an architecture where a remote user can control and interact with their IoT devices remotely via the web or mobile app (e.g., a classic example of a person *remotely* turning on or off a smart light bulb in their LAN network); or (2) the IoT device can be used as *means* to enable communication between two people (e.g., a child with a smart toy can send a voicemail to their parent's phone as in the case of the Toymail `talkie` or a parent can send a text message to their child's device and the device prints the message and physically hands it to the child such as the Turtle Mail connected toy [8]).

Fig. 2b shows this architectural pattern. As shown in the figure, all network traffic between the device and the mobile device goes through the cloud. The cloud is used as a "proxy" to enable the communication between the IoT device (deployed behind a home router and firewall) and a mobile device (running the app and that is not necessarily connected to the same network as the IoT device).

So, the user may still physically interact (directly or indirectly) with the device while other users may interact with the IoT device remotely via an app.

2.2.2 Communication Technology

We briefly describe the underlying technologies each device uses to transmit the voice of a child (captured by the microphone on the toy device) to the cloud. We found that the device may either open a continuous connection with the cloud; or establish a new connection each time the device wants to send voice message.

Continuous audio stream: both the `dino` and `barbie` open a continuous stream when the device is turned on. The `dino` device uses the session initiation protocol (`SIP`) for initiating an encrypted Voice-over-IP (`VoIP`) session between the device and the cloud. Then, the traffic is transported through the Real-time Transport Protocol (`RTP`) over `UDP`, and the traffic is encrypted at the `RTP` protocol level. This implementation is commonly used in audio or video streaming applications.

The `barbie` device also establishes a continuous session with the cloud; however, it uses a more traditional approach found in web applications. The device first establishes a secure Transport Layer Security (`TLS`) connection when the device is turned on; and the audio is constantly sent and received through `http` via `TLS`. Even when the child is not speaking anything to the device, we can still see traffic activity. (It has been shown [18] that `barbie` devices were vulnerable to the `POODLE`—padding oracle on downgraded legacy encryption—crypto bug that allows a man-in-the-middle attacker to break the `https` encryption).

Noncontinuous audio transfer: the `talkie` uses the Electric Imp technology to transfer an audio message to a `talkie` device or mobile app. The communication happens by connecting to an agent app deployed in the Electric Imp cloud [6]. Electric Imp developers claim this communication happens over encrypted `http` methods over `tcp` with full encryption [4]. One difference from the previous communication pattern is that a new connection is established when the toy wants to communicate with the cloud. For example, when a child wants to send an audio message to a friend or their parent, they first hold a button on the device, and then they speak their message. During this time the device records the audio locally, and only after the child clicks another physical button on the device, the device establishes a new connection to the back-end cloud and starts transferring the voice audio (until then, there is no traffic activity) [2]. Once the device finishes transferring, any open port is closed. Further, communication to the device is initiated only by the device itself. So, the device does not accept any incoming traffic even from devices on the same network, let alone from remote devices.

3 Threat Model

We assume that the attacker (1) acts like a man-in-the-middle in the network, (2) is able to remotely capture the network traffic between a device and the cloud, and (3) can record and replay network packets. Fig. 2 shows the attack points we consider in this paper.

In addition to a network-based attacker, we also consider an attacker that can talk to the device. This means the attacker has physical access to the device, and can press the button to talk to the device and can hear the responses. In the next section we summarize what these attackers can do to extract information about the toy owner and inject voice content.

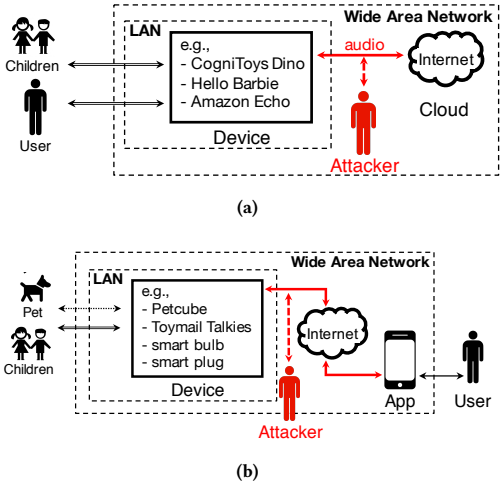


Figure 2. Attack on the network: (a) between device and cloud; or (b) either between device and cloud, or cloud and mobile device.

4 Security Analysis

We start this section by presenting our security findings on the CogniToys `dino`. We then present an example of an attack to eavesdrop on the traffic communication between the `dino` and the cloud.

4.1 Vulnerability Assessment

Overview of our findings: the privacy of the child playing with the `dino` relies on the security of the device and its communication to the cloud. In particular, if an attacker (like the one we described in Fig. 2a) is able to compromise the communication channel, then the privacy of a child can be exploited by remote unauthorized users in the network. In this paper we show that even though the traffic between the `dino` device and the cloud is encrypted, weakness in the encryption scheme poses a serious threat: it exposes voice content of the child using the device to an *eavesdropper* attacker. Further, a remote attacker is able to exploit weakness in the encryption implementation to *inject* voice content for the device to speak to a child. In a different attack, a person in possession of the device (e.g., if the device is lost or resold) can ask the `dino` invasive questions about the previous owner, and the `dino` will reply accurately, revealing privacy-sensitive information.

Physical access findings: as IoT devices become smarter and provide new ways to interact with users (e.g., via voice-activated questions and answers), manufacturers should consider authentication of the person interacting with the device before revealing sensitive information. In our experiments a different user (other than the owner) asked the `dino` questions about the owner like her name, age, birthday, where she lived, and the `dino` helpfully answered all these questions. While we are impressed by the `dino`'s ability to interact intelligently with their users (by far the most "intelligent" of the devices we analyzed) these interactions also open the door to privacy invasions. This finding is related to recent work [32] where researchers are able to inject inaudible voice commands to activate voice controllable systems (such as Siri, Google Now, Alexa) and it raises the issue of how to properly authenticate the owner of a voice-activated device.

Network analysis findings: our network analysis reveals several important results. We summarize the results here, and afterwards we describe each finding in more detail. We also include the CVEs assigned based on our findings:

(1) the device uses AES-128 with ECB mode (a weak mode of encryption) to encrypt voice traffic between the device and remote server (CVE-2017-8867);
(2) the device shares a fixed small pool of hard-coded keys (with other dino devices) to encrypt/decrypt VoIP traffic (CVE-2017-8866);
(3) the device does not provide sufficient protections against capture-replay attacks as the device accepts any voice traffic sent to them via RTP (CVE-2017-8865).

Background: the device uses the session initiation protocol (SIP) for initiating an encrypted Voice-over-IP (VoIP) communication session between the device and the cloud. As we show a snippet of the SIP traffic in Listing 2, the plain-text signaling traffic can be used to learn information about the username, crypto used, and encryption key (highlighted in red). Note that this by itself is not a vulnerability, because this is how the SIP protocol is intended to be used; and even when the attacker does not have access to this initial part of the traffic initialization, it is still possible to launch the replay-attacks we explain in the next section.

Once the voice traffic is initialized, the traffic is transported between the dino and the cloud via the Real-time Transport Protocol (RTP) over UDP. Even when there is no activity between the dino and the cloud (because the child is not speaking to the dino or the dino is not saying anything back to the child), there is a RTP connection open at all times.

We found that the voice communication between the dino and the remote server is encrypted using the AES symmetric algorithm with a 128-size key and over the insecure electronic codebook (ECB) mode which is a well-known weak mode of encryption. In addition, we observed that each time the dino initializes a new session with the cloud, the device agrees on a *key index* denoted as k=index:v and each key index is further associated with the same AES key (unknown to the attacker). The Listing 2 (mentioned earlier) shows the key index as the k value in the SIP traffic.

Listing 2. Plaintext snippets of SIP traffic reveals important information about the encryption algorithm and keys used.

```
INVITE sip:1000@[redacted];transport=tcp SIP/2.0
Via: SIP/2.0/TCP 192.168.2.5:5020
From: <sip:[redacted]@[redacted]>;tag=0JKAwIe
To: <sip:1000@[redacted]>
Call-ID: Z2iZfmNgNSd01Lx
Contact: <sip:[redact]@192.168.2.5:5020;transport=tcp>
Allow: INVITE, ACK, BYE, UPDATE
Proxy-Authorization: [redacted]
Content-Type: application/sdp
Content-Length: 205


v=0
o=xxxxxx xxxxxx xxxxxx IN IP4 [redacted]
s=Dino Call
c=IN IP4 [redacted]
t=0 0
m=audio 42806 RTP/AVP 9
a=sendrecv
a=rtpmap:9 G722/8000
a=direction:active
a=crypto:AES_128_EBC
k=index:15
```

Then, we observed that the encrypted traffic reveals a 16-byte (128-bit) data pattern that is periodically repeated in the payload, and that an attacker can further use this pattern to deterministically map encrypted voice streams to one of the AES key indexes (as we show later, this is important because an attacker can use this information to successfully launch replay-attacks, to eavesdrop, or inject voice content).

Listing 3 shows a snippet of two RTP traffic streams initialized under two different sessions, and using two different key indexes. The traffic on the left is initialized under $v = 0$, and the traffic on the right under $v = 15$. Although, both streams are initialized under two different key indexes; we can see a 16-byte pattern being repeated. For example, the hexadecimal bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105 is repeated multiple times in the stream in the left, and similarly db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e is repeated in the stream in the right. We observe that these patterns happen at the same point in time in both RTP streams, when the traffic is first initialized and there is no voice communication happening between the dino device and the cloud. We still see these patterns being repeated when the device starts sending voice traffic to the back-end cloud services, although they might not exactly align between two different streams in the same point in time as it did before (because the voice content is not precisely the same).

Further, we noticed that two traffic streams that are initialized using the same k=index:v contain the *exact same* 16-byte payload pattern that is periodically repeated throughout the encrypted traffic (i.e., every time the traffic is initialized using $v = 0$ the repeated pattern is bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105). It follows then that each data pattern directly maps to a key index v that was used during the session initialization. When traffic is initialized under the same k=index:v, then both traffic are identical when neither the child nor the dino are speaking. When either one speaks, then the traffic is different except for the 16-byte pattern that is still periodically repeated and exactly the same across the two traffic streams.

Note that if an attacker has access to a dino device, then they can use this information to initialize the device under different v values and learn a mapping between v values and the 16-byte pattern. Then, if the attacker has access only to the RTP traffic (and not the SIP traffic), they can look for the 16-byte pattern in the encrypted traffic to figure out the v used. As we show later, these observations can be used to launch attacks against dino devices.

Finding 1 - Each device uses 16 keys: we found that the device uses a fixed pool of 16 hard-coded keys to encrypt/decrypt the VoIP traffic. The key is picked at random (out of only 16 possibilities) during session initialization between the device and remote server. In particular, the key index v we mentioned before (that is passed in the SIP traffic to the remote server) corresponds to one of 16 possible AES symmetric keys.

Finding 2 - The attacker can buy a toy to decrypt traffic from a target: furthermore, we found that different dino devices share this same set of keys. Therefore RTP traffic that is encrypted by one dino device *can* be decrypted by another dino device under the condition that both devices are initialized using the same key index v value (and thus, the same key).

An attacker with this knowledge can buy another dino device and use the device as their attack listening tool: to decrypt encrypted RTP traffic from a legitimate user, and further *eavesdrop* on the traffic that was encrypted by any other dino device. Since there are only 16 possible keys, an attacker can start their dino device under the same session initialization key index that matches the same key index on the encrypted traffic by the legitimate user. All the attacker needs to do, is to keep on restarting their device until the key index in the current session matches to the one used in the

Listing 3. Similar payload patterns due to Electronic Codebook (ECB) encryption mode used.

```
$diff -y <(xxd traffic1-INDEX-0.raw) <(xxd traffic2-INDEX-15.raw) | head -n 17

0000000: bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105  .........[.  | 0000000: db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e  ......q.Z.J
0000010: bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105  .........[.  | 0000010: db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e  ......q.Z.J
0000020: 9950 ae84 56c5 c07b 2b02 1c86 bdb1 d398  .P..V..{+..  | 0000020: 5afd 508e df7e 5901 80b8 d180 aabc 6f43  Z.P..~Y.....
0000030: f9e7 74c4 1e62 263c 7aca a658 3763 60d5  ..t..b&<z..  | 0000030: 2911 bbe6 12d1 27ec efd1 3fc9 6462 140f  ).....'...?
0000040: 1bda a54a 46cc 128b 1a58 0161 0ac1 198b  ...JF....X.  | 0000040: 09a3 5749 70a7 e36b 32df c772 9050 a8c7  ..WIp..k2..
0000050: 240b 7824 b4ed 2e2d 1fb0 c24d 1421 4515  $.x$...-...  | 0000050: 76be 4c13 37ff 5b5a 1f71 a196 d306 1af3  v.L.7.[Z.q.
0000060: 0ee0 45c5 74ab 7ce0 9b7d 7636 d096 f2f4  ..E.t.|..}v  | 0000060: ec76 5b15 97c9 13a7 b5e7 8bd2 6f16 b848  .v[........
0000070: 96f1 90e0 c8dd beac 8de6 011d ec6c 5ac6  .........    | 0000070: 7ef5 6154 fe34 8c1d 8fc5 30c0 8658 66d4  ~.aT.4....0
0000080: 262a 1a16 bfa1 06df f07c 32df c4c1 e06b  &*.....|2    | 0000080: c6db 2668 d3a7 0d90 462e 4898 e800 3b13  ..&h....F.H
0000090: 5ae5 2e83 9d2a cd1a 1282 66e9 eeab b65f  Z....*....f  | 0000090: 9f71 afeb bc31 1475 d25b 1d05 13e8 45b1  .q...1.u.[.
00000a0: bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105  .........[.  | 00000a0: db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e  ......q.Z.J
00000b0: bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105  .........[.  | 00000b0: db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e  ......q.Z.J
00000c0: bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105  .........[.  | 00000c0: db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e  ......q.Z.J
00000d0: bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105  .........[.  | 00000d0: db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e  ......q.Z.J
00000e0: bbc7 2ed6 b4dd dbbe c35b d9fd 8018 5105  .........[.  | 00000e0: db86 fddb a1f2 71c1 5ad6 4a93 9a18 b01e  ......q.Z.J
00000f0: d731 9c8f c482 ddf1 f18d 856b 55b6 6cf1  .1.........  | 00000f0: c03f 5e67 9fb5 3d7c d926 7861 9d1f 0a39  .?^g..=|.&x
0000100: 394c e43b 4a0f 0527 b2d0 6ed5 ea98 7f30  9L.;J..'..n  | 0000100: 9685 bd44 0cc0 c7d6 3d2a 0f8f 9cd7 6c66  ...D....=*.
```

encrypted traffic the attacker wants to eavesdrop (e.g., RTP traffic the attacker intercepted in a remote network). When the attacker is able to capture encrypted traffic of a victim, and establish a session between the attacker's dino device and the cloud under the same key index (and therefore, same key), then any encrypted traffic can be decrypted by the attacker's device.

Finding 3 - The attacker can inject audio in a target device: a man-in-the-middle remote attacker on the network can record VoIP traffic between its device and the remote server, and replay that traffic to a target dino device, which in turn will accept voice traffic sent to them via RTP, and attempt to play out loud the voice stream on the device's embedded speakers without any type of authentication.

4.2 Attack Details

Based on the findings we described, we now present one of the attacks we tested with success on dino devices. Here we show that it is possible to eavesdrop on encrypted traffic captured between one dino device and the cloud by replaying the traffic to another dino device and hearing the child's voice coming out from the embedded speakers of the second dino device. We also tested the audio injection attack against dino devices, but because of space constraints (and because the attack follows a similar methodology to the eavesdropping attack) we will publish the injection attack in an extended version of this paper.

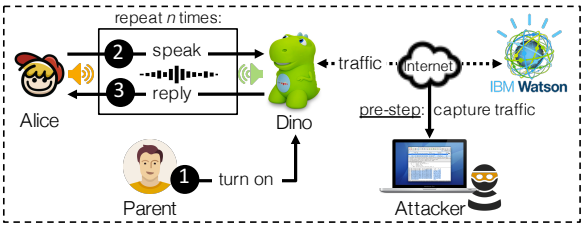
We divide this attack into two phases: (1) traffic capturing and (2) eavesdropping. During the traffic capturing phase, an attacker captures the encrypted network traffic that they are interested in listening; and during the eavesdropping phase, the attack launches a *replay-attack* to decrypt and play-back the voice traffic on the attacker's own dino.

The traffic capturing phase is straightforward and illustrated in Fig. 3a. So we focus on describing the second phase.

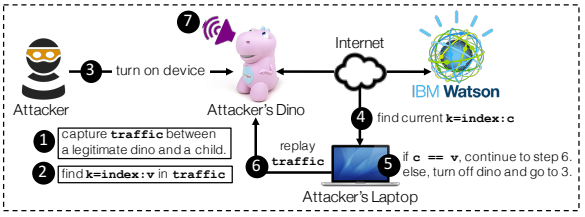
Eavesdropping phase: a malicious user launches replay-attacks to successfully play out loud the child's voice to speakers embedded on their own dino device. Fig. 3b illustrates this phase as follows:

Step 1: We assume the attacker captured RTP (encrypted) traffic between a child's dino and the cloud, and now wants to hear the voice content (but does not have the private keys).

Step 2: The attacker finds out what key index v was used to encrypt the gathered traffic that the attacker wants to listen:
- **2a:** The attacker looks at the v value in the SIP traffic; or
- **2b:** The attacker maps the repeated 16-byte pattern in the encrypted RTP stream to v. This mapping is predictable.



(a) Traffic capturing phase



(b) Eavesdropping phase

Figure 3. (a) attacker taps on the RTP traffic between child's dino and cloud, and captures the network packets; (b) attacker is able to clearly listen the voice conversation between a child and the child's dino. This is done by replaying the captured encrypted traffic from the legitimate user using the dino device, to the attacker's dino.

Step 3: The attacker turns on their attacker's dino device.

Step 4: The attacker finds out what key index "c" was used during the initialization session of the attacker's dino device.

Step 5: The attacker checks if $c == v$; that is, if the gathered RTP traffic and the current attacker's dino session use the same key index (and moreover, the same AES key!). If false, restart attacker's dino and go back to step 3; otherwise, go to step 6.

Repeat steps 3-5 until the attacker's dino is initialized under the same v value. When this happens, then both the captured RTP traffic (from one device) and the current session (from another device) share the same AES-128 key. As we show next, then the attacker's dino is able to decrypt and play-back the encrypted RTP traffic captured from a child's dino.

Step 6: As shown in Fig. 3b, the attacker's computer is placed as a man-in-the-middle between the cloud and attacker's dino:

6.1: Attacker rewrites RTP traffic (from previous phase) to appear as if it was coming from cloud to attacker's dino; e.g., modify source and destination ports, IP addresses, etc.

6.2: Attacker *replays* the modified RTP traffic to attacker's dino using the machine placed between attacker's dino and the cloud with a session established under $c == v$.

Step 7: The attacker's dino receives the RTP traffic sent by the attacker as if it was coming from the cloud, and the attacker hears the child's voice coming from the speakers of the attacker's dino.

We can use a similar approach as the one described here to inject voice content to a child's device. In this case, an attacker would need to speak to their own device the content they want to inject to a child's device (under all possible 16 encryption keys) and save the network packets; and then intercept the communication to a child's dino device and perform the replay attack using one of the traffic streams captured (that was encrypted under the same key index as the current session between the child's dino and the cloud).

Since both the eavesdropping and audio injection attacks are performed by injecting RTP traffic directly into the network traffic communication to a dino device (to appear as if the RTP traffic was coming from the cloud), it is not possible for the remote server to ever notice that these attacks are happening. Further, because the device does not authenticate incoming voice traffic (it accepts RTP traffic from any destination as long as it comes to the correct port under the current established connection), it is also not possible for the toy to notice that the RTP traffic is coming from a man-in-the-middle attacker. One way to solve this is to instead of using the RTP protocol use the Secure Real-time Transport Protocol (SRTP) [13] which provides message authentication and integrity checking, and protections against replay attacks on the RTP traffic.

5 Conclusions

In this paper we studied the security practices of three smart toys and illustrated some of their similarities and differences. Understanding the recurrent patterns that developers use to design these toys can help us identify new unanticipated vulnerabilities, and propose new solutions.

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. For example, the attacker can inject audio to the device so the smart toy tells children to open the door to their houses, or to change combination locks, or even tell them lies about their parents or other subjects the attacker may find useful.

Our physical attack also uncovered the problem of authenticating the user of voice-activated devices. Recent work [32] has shown that it is possible to inject inaudible voice commands to activate services like Siri, Google Now, and Alexa, even when the device authenticates the user (the authors were able to construct wake up words by extracting phonemes from a voice recording of victim users). So this is still an open research problem.

Vulnerability Disclosure

Per US-CERT's recommendation, we disclosed the vulnerabilities we found directly to CogniToys on March 8, 2017 and they replied promptly. We are currently working with them to address these issues, and at the time of this writing the vendor was testing an updated firmware to address the problems we found.

Acknowledgments

This material is based upon work supported by NSF under award number CNS 1553683.

References

[1] [n. d.]. Electric Imp - Connect your products quickly and securely. https://electricimp.com/. ([n. d.]).
[2] [n. d.]. Electric Imp - Effective Internet-agent-device Communication. https://electricimp.com/docs/resources/interactive/. ([n. d.]).
[3] [n. d.]. Electric Imp BlinkUp. https://electricimp.com/platform/blinkup/. ([n. d.]).
[4] [n. d.]. Extra protection for agents that host APIs. https://electricimp.com/docs/resources/agentsecurity/. ([n. d.]).
[5] [n. d.]. Meet Hello Barbie. http://hellobarbiefaq.mattel.com. ([n. d.]).
[6] [n. d.]. Network Requirements for imp-enabled Devices. https://electricimp.com/docs/troubleshooting/networks. ([n. d.]).
[7] [n. d.]. Toymail - Stay in touch with your kids! https://toymail.co/. ([n. d.]).
[8] [n. d.]. Turtle Mail. https://aedreams.com/shop/. ([n. d.]).
[9] 2017. Boy, 11, hacks cyber-security audience to give lesson on 'weaponisation' of toys. https://www.theguardian.com/world/2017/may/17/boy-11-hacks-cyber-security-audience-to-give-lesson-on-weaponisation-of-toys. (May 2017).
[10] 2017. CC3200 SDK Sample Applications. http://processors.wiki.ti.com/index.php/CC3200$_{S}$DK$_{S}$ample$_{A}$pplications. (May 2017).
[11] 2017. From Barbies to bears, interactive 'smart toys' could be a hacker's plaything. https://au.news.yahoo.com/a/36421102/from-barbies-to-bears-interactive-toys-could-be-target-for-hacke/. (July 2017).
[12] Michael Bahar. 2017. The FTC is watching when your children's toys are listening. http://thehill.com/blogs/pundits-blog/technology/344554-the-ftc-is-watching-when-your-childrens-toys-are-listening. (Aug. 2017).
[13] Mark Baugher, D McGrew, M Naslund, E Carrara, and Karl Norrman. 2004. *The secure real-time transport protocol (SRTP).* Technical Report.
[14] Thomas Claburn. December 8, 2016. Playtime's over: Internet-connected kids toys 'fail miserably' at privacy. http://www.theregister.co.uk/AMP/2016/12/08/connected$_{t}$oys$_{f}$ail$_{m}$iserably$_{a}$t$_{p}$rivacy/. (December 8, 2016). Accessed: Jan 2017.
[15] CogniToys. [n. d.]. Meet the CogniToys Dino. https://cognitoys.com/. ([n. d.]).
[16] Luke Cooper. 2017. 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. (Feb. 2017).
[17] Samuel Gibbs. 2015. Privacy fears over 'smart' Barbie that can listen to your kids. https://www.theguardian.com/technology/2015/mar/13/smart-barbie-that-can-listen-to-your-kids-privacy-fears-mattel. (2015).
[18] Dan Goodin. 2015. Internet-connected Hello Barbie doll gets bitten by nasty POODLE crypto bug. https://arstechnica.com/information-technology/2015/12/internet-connected-hello-barbie-doll-gets-bitten-by-nasty-poodle-crypto-bug/. (Dec. 2015).
[19] Laura Hautala. 2017. Smart toy flaws make hacking kids' info child's play. https://www.cnet.com/news/cloudpets-iot-smart-toy-flaws-hacking-kids-info-children-cybersecurity/. (Feb. 2017).
[20] Patrick CK Hung, Marcelo Fantinato, and Laura Rafferty. 2016. A Study of Privacy Requirements for Smart toys. In *PACIS*. 71.
[21] Emily McReynolds, Sarah Hubbard, Timothy Lau, Aditya Saraf, Maya Cakmak, and Franziska Roesner. 2017. Toys That Listen: A Study of Parents, Children, and Internet-Connected Toys. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 5197–5207.
[22] Kevin Meurer et al. 2016. Can (and Should) Hello Barbie Keep a Secret? (2016).
[23] Tim Moynihan. 2015. This Toy Dinosaur Uses IBM's Watson as a Brain. https://www.wired.com/2015/08/toy-dinosaur-uses-ibms-watson-brain/. (Aug. 2015).
[24] Mike Murphy. 2017. Don't buy your kids Internet-connected toys. https://qz.com/920482/dont-buy-your-kids-internet-connected-toys/. (Feb. 2017).
[25] Jared Newman. 2015. Internet-connected Hello Barbie doll can be hacked. http://www.pcworld.com/article/3012220/security/internet-connected-hello-barbie-doll-can-be-hacked.html. (Dec. 2015).
[26] Kari Paul. 2017. Read this before buying a Wi-Fi-connected toy for your child. http://www.marketwatch.com/story/dont-buy-a-wi-fi-connected-toy-for-your-child-without-reading-this-2017-07-20. (July 2017).
[27] Gil Reiter. 2014. A primer to Wi-Fi® provisioning for IoT applications. In *Texas Instruments White Paper*.
[28] April Glaser. Slate. 2017. The FBI Is Warning Parents About the Risks of Internet-Connected Toys Spying on Kids. (July 2017).
[29] Emmeline Taylor and Katina Michael. 2016. Smart Toys that are the Stuff of Nightmares. *IEEE Technology and Society Magazine* 35, 1 (2016), 8–10.
[30] Texas Instruments. [n. d.]. Overview for SimpleLink CC3x family of wireless MCUs. https://www.ti.com/lsds/ti/microcontrollers$_{1}$6-bit$_{3}$2-bit/wireless$_{m}$cus/simplelink$_{c}$c3x/overview.page. ([n. d.]).
[31] Shaun Waterman. 2017. FTC pushed from Hill on hacking of smart toys, kids' privacy. https://www.cyberscoop.com/ftc-pushed-hill-hacking-smart-toys-kids-privacy/. (May 2017).
[32] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. DolphinAttack: Inaudible Voice Commands. In *Proceedings of ACM Conference on Computer and Communications Security (CCS'17)*.