Parental Controls: Safer Internet Solutions or New Pitfalls?

Suzan Ali, Mounir Elgharabawy, Quentin Duchaussoy, Mohammad Mannan, and Amr Youssef | Concordia University

Parental-control solutions often require dangerous privileges to function. We analyzed privacy/security risks of popular solutions and found that many leak personal information and are vulnerable to attacks, betraying the trust of parents and children.

Many children are now as connected to the Internet as adults are, if not more. The Internet provides an important avenue for education, entertainment, and social connection for children. However, the dark sides are also significant: Children are by nature vulnerable to online exploitation, Internet addiction, and other negative effects of online social networking, including cyberbullying and even cybercrimes. To provide a safe Internet experience, many parents rely on parental-control solutions, which are also recommended by government agencies, including the U.S. Federal Trade Commission (FTC) and the U.K. Council for Child Internet Safety.

Parental-control solutions are available for different platforms, including desktop applications, browser extensions, mobile apps, and network devices that can monitor all connected computers and smart devices. Most of these solutions require special privileges to operate, such as mobile device administration/management capabilities, Transport Layer Security (TLS) interception, access to browsing data, and control over the network traffic. In addition, they also collect a lot of sensitive user data, such as voice, video, location, messages, and social media activities. Thus, design and implementation flaws in these solutions can lead to serious privacy leakage and online and real-world security and safety issues.

To better understand the privacy and security implications of parental-control solutions, we designed an experimental framework with a set of security and privacy tests and systematically analyzed popular representative solutions: eight network devices, eight Windows applications, 10 Chrome extensions, and 46 Android apps representing 28 Android solutions, grouped by vendor (an Android solution is typically composed of a child app, a parent app, and an online parental dashboard). We found 170 vulnerabilities in the tested solutions; the majority of solutions broadly fail to adequately preserve the security and privacy of

Digital Object Identifier 10.1109/MSEC.2021.3076150
Date of current version: 19 May 2021

both children and parent users. Our notable findings include:

- The Blocksi parental-control router allows remote command injection, enabling an attacker with a parent's email address to eavesdrop and modify the home network's traffic or use the device in a botnet (for example, Mirai). Blocksi's firmware-update mechanism is also completely vulnerable to network attackers.
- Nine out of 28 Android solutions and four out of eight network devices do not properly authenticate their server application programming interface (API) endpoints, allowing illegitimate parties to access and view/modify server-stored children/parent data.
- Six out of 28 Android solutions allow an attacker to easily compromise the parent account at the server end, enabling full account control of the child's device (for example, the attacker can install/remove apps and allow/block phone calls and Internet connections).
- Eight out of 28 Android solutions transmit personally identifiable information (PII) via HTTP (for example, kidSAFE-certified Kidoz sends account credentials via HTTP).

As part of responsible disclosure, we shared our findings and possible fixes with all of the solution providers. Two months after disclosure, only ten companies responded, with seven custom and three automatic replies. Notable changes after the disclosure include: MMGuardian deprecated their custom browser, FamiSafe fixed the Firebase database security issue, and FamilyTime enabled HTTP Strict Transport Security (HSTS) on their server. Details of our findings and disclosure responses are available in the Annual Computer Security Applications Conference version of our article.[7]

Related Work

Over the past years, several parental-control tools have made the news for security and privacy breaches. Example exposures include when TeenSafe leaked thousands of children's Apple IDs and passwords and when Family Orbit exposed nearly 281 gigabytes of children's photos and videos on a cloud server.

Between 2015 and 2017, researchers from the Citizen Lab (citizenlab.ca), Cure53 (cure53.de), and OpenNet Korea (opennetkorea.org) published a series of technical audits[1] mandated by the Korean government of three popular Korean parenting apps, revealing serious security and privacy issues in them. In 2019, Feal et al.[2] studied 46 parental-control Android apps for data collection and data-sharing practices and the completeness and correctness of their privacy policies. In some of these apps, we further identified new critical security issues (for example, the leakage of plaintext authentication information) using our comprehensive app-analysis framework. Reyes et al.[3] analyzed children's Android apps for Children's Online Privacy Protection Act (COPPA) compliance. Out of 5,855 analyzed apps, the majority of them were found to potentially violate COPPA, and 19% were found to send PII in their network traces. Our analysis across multiple platforms is inspired by existing work and past security incidents, and it provides a broader picture of the security and privacy risks of parental-control tools.

Background and Threat Model

Monitoring Techniques

Network parental-control devices can monitor network traffic but usually cannot inspect the content of encrypted traffic. The analyzed devices act as man-in-the-middles (MITMs) between the client device and the Internet router by performing Address Resolution Protocol (ARP) spoofing or by creating a separate access point (AP) for all children's devices. ARP spoofing enables the network device to impersonate the home router and monitor all of the local network traffic.

Android apps rely on several Android-specific mechanisms, including the following:

- *device administration*: provides several administrative features at the system level, including device lock, factory reset, certificate installation, and device-storage encryption
- *mobile device management*: enables additional control and monitoring features and is designed for businesses to fully control/deploy devices in an enterprise setting
- *Android accessibility service*: enables the capturing and retrieving of window content, logging keystrokes, and controlling website content by injecting JavaScript code into visited web pages
- *Android virtual private network, custom browsers, and third-party domain classifiers:* used to filter web content
- *access to Facebook and YouTube OAuth credentials*: used to monitor a child's activities on Facebook and YouTube.

Windows applications use the following techniques: a TLS proxy is installed by inserting a self-signed certificate in the trusted root certificate store, allowing content HTTPS content analysis/modification; user applications are monitored for usage and duration; and user activity is monitored via screenshots, keylogging, and webcam access. Parental-control Chrome extensions use Chrome APIs to monitor the user-requested

uniform resource locators (URLs), which includes intercepting and redirecting traffic and modifying page content and metadata, including cookies.

Threat Model

We consider the following attacker types with varying capabilities but that require no physical access to either a child/parent's device or back-end servers:

- *on-device attacker*: a malicious app with limited permissions on a child/parent's device
- *local network attacker*: an attacker with direct or remote access to the same local network as a child's device
- *on-path attacker*: an MITM attacker between the home network and a solution's back-end server
- *remote attacker*: any attacker who can connect to a solution's back-end server.

Potential Security and Privacy Issues

We define the following list of potential security and privacy issues to evaluate parental-control tools (tested using only our own accounts where applicable). This list was initially inspired by previous work[1,4–6] and then was iteratively refined by us.

1. *Vulnerable client product*: This is a parental-control product (including its update mechanism) being vulnerable, allowing sensitive-information disclosure (for example, via on-device side channels) or even full product compromise (for example, via arbitrary code execution).
2. *Vulnerable back end*: This is the use of remotely exploitable outdated server software and misconfigured or unauthenticated back-end API endpoints (for example, Google Firebase in Android apps).
3. *Improper access control*: This is the failure to properly check whether the requester owns the account before accepting queries at the server end (for example, insecure direct object reference).
4. *Insecure authentication secrets*: This is the plaintext storage or transmission of authentication secrets (for example, passwords and session IDs).
5. *Secure socket layer (SSL) Strip attack*: A parental-control tool online management interface is vulnerable to SSLStrip attacks that strip away the security provided by HTTPS, exposing private information in plaintext [countermeasures exist (for example, HSTS) but must be correctly implemented].
6. *Weak password policy*: Very weak passwords (for example, with four characters or fewer) are accepted.
7. *Online password brute force*: There is no defense against unlimited login attempts on the online parental-login interface (for example, the Completely Automated Public Turing Test to Tell Computers and Humans Apart).
8. *Uninformed suspicious activities*: There are no notifications to parents about indicators of possible compromise (for example, the use of parental accounts on a new device or password changes).
9. *Insecure PII transmission*: This is the sending of PII from the client end without encryption, allowing an adversary to eavesdrop for PII.
10. *PII exposure to third parties*: This is the direct PII collection and sharing (from client devices) with third parties.

Selection of Parental-Control Solutions

We chose solutions used in the most popular computing platforms for mobile devices (Android), personal computers (Windows), web browsers (Chrome), and selected network products from popular online marketplaces (Amazon). We used "Parental Control" as a search term on Amazon and Chrome Web Store and selected eight devices and ten extensions. For Windows applications, we relied on rankings and reviews provided by specialized media outlets and selected eight applications.

We selected 158 apps with more than 10,000 installations from Google Play, four companion apps for network devices, and six additional apps available on their official websites with additional features—making the total 153 (after removing 15 unresponsive/unrelated apps). Fifty one of these are purely children's apps; 24 are purely parents' apps; and 78 are used for both parents' and children's devices. For an in-depth analysis, we picked 46 popular Android apps representing 28 parental-control solutions.

Methodology

We combined dynamic (primarily traffic and usage) and static (primarily code review/reverse-engineering) analyses to identify security and privacy flaws in parental-control tools (for an overview, see Figure 1). For each product, we first conducted a dynamic analysis and captured the parental-control tool's traffic during its usage (as parents/children). If the traffic was in plaintext or decryptable (for example, via TLS interception), we also analyzed the sent information. Second, we statically analyzed their binaries (via reverse engineering) and scripts (if available). We paid specific attention to the API requests and URLs that were present in the code to complement the dynamic analysis. After merging the findings, we looked into the contacted domains and checked the traffic for security flaws (for example, TLS weaknesses). Third, we tested the security and privacy issues listed under "Potential Security and Privacy Issues" against the collected API URLs and requests.

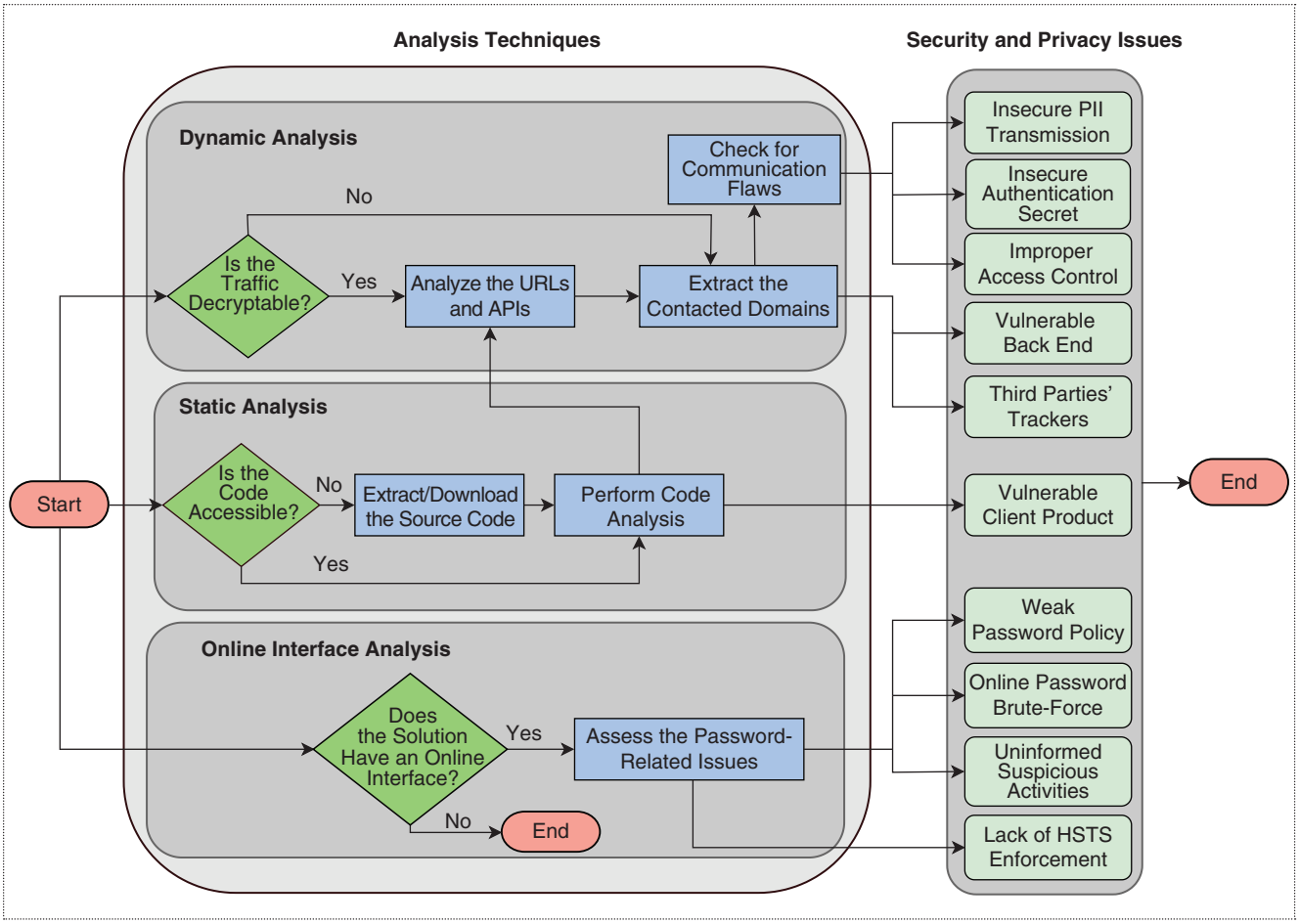

Figure 1. An overview of our evaluation framework.

For the parental-control tools with online interfaces, we assessed the password-related issues and tested the SSLStrip attacks against the login pages.

Dynamic Analysis

In this section, we summarize our dynamic analysis experiments, including the analysis we perform on our collected traffic, and the back-end server software of each solution.

Usage Emulation and Experimental Setup. We set up test environments for each solution, emulating user actions over the span of hours to days, with the goal of triggering user interface (UI) events and looking for signs of PII leakage, weak security measures, or potential vulnerabilities. We collected the traffic from the children's, parents', and network devices and then performed relevant analysis. We evaluated each web-filtering mechanism by visiting a blocked website (gambling/adult) and a university website. We also performed user activities monitored by platform-specific parental-control features and evaluated the solutions' operations. For example, on Android, we performed basic phone activities [short message services (SMSs) and phone calls] and Internet activities (instant messaging, social media, browsing, and accessing blocked content).

We evaluated the network devices in a lab environment by connecting them to an Internet-enabled router (like in a domestic network setup) with the OpenWrt firmware. We used test devices with web browsing to emulate a child's device. If the parental-control device used ARP spoofing, the test device was connected directly to the router's wireless AP; otherwise, the test device was connected to the parental-control device's wireless AP. We captured network traffic on both the test device and router using Wireshark and tcpdump, respectively.

For Android apps, we used separate Android phones to concurrently record and inspect network traffic originating from the children and parents' apps. We tested each Windows application and Chrome extension on a fresh Windows 10 virtual machine with Chrome and mitmproxy installed.

Traffic Analysis. After intercepting traffic, we parsed and committed the collected traffic to an SQLite database and checked for the following security and privacy-related issues: We checked for PII and authentication secrets transmitted in plaintext or leakage of PII to third-party domains. We automatically searched for PII items (that is, case-insensitive partial string match) in the collected traffic and recorded the leaked information, including the HTTP-request URL. We decoded the collected network traffic using common encoding (base64 and URL) and encoded possible PII using hashing algorithms (MD5, SHA1, SHA256, and SHA512) to find out obfuscated leaks.

To find API endpoints with improper access control, we first identified all of the APIs that could potentially be exploited (without strong authentication) by replaying the recorded HTTP request stripped of authentication headers (for example, cookies and authorization headers). Then, we retrieved the parameters used by these APIs (for example, keys, tokens, or unique IDs) and assessed the parameters in terms of their predictability and confidentiality. We compiled lists of known trackers and then identified communication to these trackers and other third-party software development kits (SDKs) in the parental-control tools' traffic (third-parties are defined as any domain other than the product providers).

Back-End Assessment. We only looked into the back ends' software components that were disclosed by web servers or frameworks in HTTP response headers, such as "Server" and "X-Powered-By." We then matched these components against the common vulnerabilities and exposures database to detect known vulnerabilities associated with these versions.

Static Analysis

Our static analysis aimed to complement the dynamic analysis whenever we could not decrypt the network traffic (for example, in the case of network devices using TLS). We used static analysis to identify PII leakage, contacted domains, weak security measures (for example, bad input sanitization), or potential flaws in implemented mechanisms.

We analyzed the network device firmware whenever possible. We attempted to either extract the firmware directly from the device (via physical interfaces) or download the device firmware from the vendor's website. We then scanned the network devices with several tools (OpenVas, Nmap, Nikto, and Routersploit) and matched the identified software versions against public vulnerability databases.

We manually analyzed the source code of the Chrome extensions, which mainly consists of scripts, separated into content scripts and background scripts. We performed an automated analysis on all of the 153 Android apps using the Firebase Scanner (github.com/shivsahni/FireBaseScanner) to detect security misconfigurations in Firebase (a widely used back-end infrastructure management for Android apps). We also used LibScout (github.com/reddr/LibScout) to identify third-party libraries embedded in these apps. Since LibScout does not distinguish which libraries are used for tracking purposes, we used Exodus-Privacy (reports.exodus-privacy.eu.org/en/trackers/) to classify tracking SDKs. We used MOBSF (github.com/MobSF/Mobile-Security-Framework-MobSF) to extract the list of third-party tracking SDKs from all of the 153 apps based on Exodus-Privacy's tracker list.

Online Interface Analysis

The online UI is the primary communication channel between parents and parental-control tools. It displays most of the data collected by the solutions and may remotely enable more intrusive features. Compromising a parent's account can be very damaging, and, thus, we evaluated the security of this interface. To check for SSLStrip attacks, we first set up a Wi-Fi AP with a set of network-interception tools installed. Then, we connected the parental-control tools to our Wi-Fi AP and launched the SSLStrip script (github.com/moxie0/sslstrip). We confirmed the effectiveness of an attack by comparing the result to the corresponding traffic in a regular testing environment. To test a password policy, we checked if a service would accept a password with four characters or less. We also used Burp Suite (portswigger.net/burp) to perform password brute-force attacks and to limit our script to only 50 authentication attempts on our own account from a single computer. We also tested two scenarios in which a parent should be notified (for example, via email): the modification of the user's password and a connection to the account from a new/unknown device.

Results

We analyzed the parental-control tools from March 2019 to September 2020, which include eight network devices, 46 Android apps representing 28 Android solutions, 10 Chrome extensions, and eight Windows applications. We present some of the most prominent findings on the tested security and privacy issues (further details are in Suzan et al.[7]) in Table 1.

Vulnerable Client Product

Here we summarize (selected) results of our analysis on vulnerable client products.

Network Devices. The Blocksi firmware update happens fully through HTTP. An integrity check is done on the

Table 1. The most significant results for security flaws in parental-control tools, labeled following our threat model.

Security flaw	Network devices						Android solutions														Chrome			Windows			
	Circle Home Plus	KoalaSafe	KidsWifi	Blocksi Router	HomeHalo	Fingbox	FamilyTime	FamiSafe	Kidoz	KidControl	KidsPlace	Life360	MMGuardian	MobileFence	Qustodio	ScreenTime	SecureTeen	Sentry	Safe Lagoon	Locategy	Easy parental control	Keepers Child Safety	Bosco	Safekid	Blocksi Web Filter	Porn Blocker	FamilyFriendly	Qustodio	Dr. Web	Spyrix	KidLogger
Vulnerable client product		●	◕	◑				○	◑		◑	○	◑								○								◑	◑	
Vulnerable back end	●	●	●		●	●		●					◑	◑			◑											◑			◑
Improper access control	◕	◕		●	◑		●	○		◑	◑		◑			●							●		–	–	–				
Insecure authentication secret			◕				◑		◑		◑		◑		◑		●						●		–	–	–				
SSLStrip attack	–	◕	◑	◑			◑	◑	◑	◑	◑	◑		◑	◑		◑	–	◑	◑	–	–	–			–	–	–	◑	–	◑	◑
Online password brute force			●	●			●	●	●	●		●	●	●	●	●	●	●	●	●	–	–	–	●		–	–	–	●		●	●
Weak password policy		●		●			●			●			●	●			●	●	●		–									–		●
Uninformed suspicious activities		●		●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	–			●		–	–	–	●		●	●
Insecure PII transmission	◑	◑		◑	◑					◑	◑	◑	◑	◑			◑				○				◑	◑	◑					
PII exposure to third parties	●						●	●	●	●	●	●	●	●	●	●	●	●	●	●			●	●	●	●	●	●				

○: *on-device attacker;* ◑: *local network attacker;* ◕: *on-path attacker;* ●: *remote attacker;* –: *not applicable;* `blank`: *no flaw found. In case the vulnerability can be exploited by two types of attackers, we display the fullest circle applicable.*

downloaded binary image using an unkeyed SHA256 hash (again, retrieved using HTTP), thus rendering it useless. Therefore, an on-path attacker can trivially alter the update file and inject its own malicious firmware into the device (see Figure 2).

Android Apps. We found that three out of the 28 Android solutions (FamiSafe, KidsPlace, and Life360) do not encrypt stored user data on shared external storage that can be accessed by any other apps with the permission to access the Secure Digital card. Examples of the sensitive information include a parent's email address and PIN code, phone numbers, a child's geolocation data, messages and social media chats, visited websites, and even authentication tokens, which enabled us to remotely read private information from a child's account. In addition, Kidoz, KidsPlace, and MMGuardian use custom browsers to restrict and filter web content. These three browsers fail to enforce HSTS (a security protocol designed to protect against SSLStrip attacks) and lack persistent visual indication of whether the website is served on HTTP.

Windows Applications and Chrome Extensions. Other than Kidswatch, all of the tested Windows applications relied on TLS proxies to operate. Some of these proxies do not properly perform certificate validation. For example, Qustodio and Dr. Web accepted intermediate certificates signed with SHA1, and none of the proxies rejected revoked certificates. Two Chrome extensions download and run a third-party tracking script at runtime, bypassing the static control of Chrome for extension security, which has been exploited in the wild by attackers tricking developers into adding malicious scripts masquerading as tracking scripts.

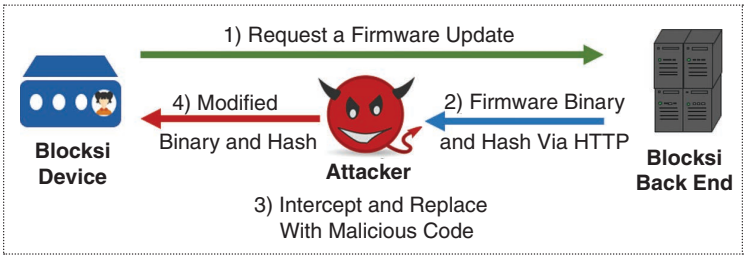

Figure 2. The Blocksi update mechanism flaw.

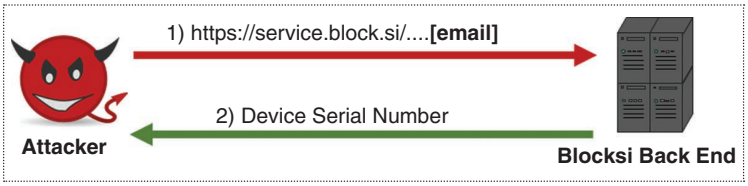

Figure 3. The Blocksi improper access control.

Vulnerable Back End

Google Firebase is a popular back-end service used in 115 out of 153 of our Android apps' data sets. Critical misconfigurations can allow attackers to retrieve all of the unprotected data stored on the cloud server. We found eight Android apps with insecure Firebase configurations. Prominent exposures include (verified using our own accounts):

- FamiSafe (with more than 500,000 installs) exposes the parent's email address.
- Locate (with more than 10,000 installs) exposes the child's name, phone number, and email address.
- My Family Online (with more than 10,000 installs) exposes the child's name, child and parent's phone numbers, parent's email address, and apps installed on the child's phone. Following our disclosure, FamiSafe fixed the Firebase security issue.

Improper Access Control

For Blocksi's login API endpoint, the device's serial number (SN) and the registered user's email address are required to authenticate the device to the server. However, a remote attacker needs to know only one of these parameters to authenticate as the attacker can retrieve a user's email address using the device SN or vice versa, thus accessing sensitive information about the home network [for example, the Wi-Fi password and media access control (MAC) addresses of the connected devices]. This information is presented in Figure 3.

Authenticating to HomeHalo's API endpoint only requires a device's SN and an HTTP header called `secretToken` (an apparently fixed value of 100,500). An on-path attacker can intercept and modify these messages and gain access to admin controls (for example, the wireless service set identifier (SSID), password, or even the device's root password).

We found that nine out of the 28 Android solutions lack authentication for accessing PII. Prominent examples include the following:

- In SecureTeen, we found an API endpoint that enables any adversary to remotely compromise any parental account by knowing only the parent's email address, allowing the attacker to monitor and control the child's device.
- In FamilyTime, a six-digit parameter `childID` is generated through a sequential counter incremented by one per user, allowing a remote attacker to collect the child's name, gender, date of birth, email address, and phone number (by simply trying all six-digit values for childID).
- In FamiSafe, an attacker app can retrieve all of the child's social media messages and YouTube activities

labeled as suspicious through an API request that requires several parameters stored in a log file on the shared external storage (available to all of the installed apps).
- Bosco's API endpoint fails to check the relation between the provided secure authorization token and the information requested, allowing any parent account with a valid token to request information about any registered child, knowing only the ID that is set to the Android advertising ID (AAID) by Bosco. However, AAID is available to all apps, thus enabling an attacker with an app on the child's device to easily retrieve the child's PII (for example, geolocation, phone call history, and pictures).

Insecure Authentication Secret

During the setup procedure of KidsWifi, the device creates an open wireless AP with SSID "set up kidswifi," making it temporarily vulnerable to eavesdropping. The parent has to use this AP's captive portal to configure the KidsWifi device to connect to the home network. As this AP is open and the client-device communication happens through HTTP, the home router's WAN and KidsWifi's local area network credentials consequentially become available to local attackers.

Kidoz exposes the user's email address and password in HTTP when the "Parental Login" link is clicked from the https://kidoz.net home page. KidsPlace and Qustodio leak session-authentication cookies via HTTP, exposing the child's current location and history of movements and the remote-control functions on the child's phone (for example, block all phone calls) in Qustodio. For KidsPlace, the attacker can lock the child's phone, disable the Internet, install malicious apps, and upload harmful content to the child's mobile.

SSLStrip and Online Account Issues

We found that 11 Android solutions, four network devices, and three Windows applications transmit the parent's account credentials via HTTP under an SSLStrip attack, potentially granting access to the parental interface for a long time. In addition, we identified that the BlueSnap.com online payment solution used by Kidoz was equally vulnerable, exposing the parent's credit card information. In Qustodio, we could extract the child's Facebook credentials provided by the parent during the configuration of the monitoring component. Following our disclosure, only FamilyTime enabled HSTS on their server. In terms of defense against online password guessing, we found that two network devices and 17 Android solutions leave their online login interfaces open to password brute-force attacks. Also, two network devices, seven Android solutions, and three Windows applications enforced a weak password policy (that is, shorter than four characters).

Insecure PII Transmission

We found that the KoalaSafe and Blocksi network devices append the child device's MAC address, firmware version number, and SN into outgoing Domain Name System (DNS) requests, allowing on-path attackers to persistently track the child's web activities.[8] HomeHalo also appends the child device's MAC address to HTTP requests to its back-end server. Several Android solutions also sent cleartext PII, including FindMyKids (the child's surrounding sounds and photo), KidControl (the parent's name and email address, geolocation, and SOS requests), and MMGuardian (the parent's email address and phone number and the child's geolocation).

Third-Party SDKs and Trackers

Some legislations (for example, U.S. COPPA and the European Union GDPR) regulate the use of third-party trackers in the services targeting children (as in, individuals under 13 years of age). We thus evaluated the potential use of third-party tracking SDKs in the parental-control tools. We found notable use of third-party SDKs in parental-control tools, except in Windows. For network devices, we identified the use of third-party SDKs in the companion apps but not in the firmware.

Trackers. We identified several tracking third-party SDKs from the network traffic generated during our dynamic analysis from the child's device. Except SecureTeen and Easy parental control, 26 out of 28 Android solutions use tracking SDKs (one to 16 unique trackers). Our traffic analysis confirms violations of COPPA—more than 30% of Android solutions utilize doubleclick.net without passing the proper COPPA-compliant parameter from the child's device. We also found that one of the network devices' companion apps, Circle, includes a third-party analytical SDK from Kochava and shares the device ID (enables tracking across apps) and device data (enables device fingerprinting for persistent tracking). To comply with COPPA, Kochava provides an opt-out option, which is not used by Circle.

Restricted SDKs From Past Work. We also studied the SDKs identified in past studies[2,3] that are restricted by their developers (for example, fully prohibited or used with particular parameters) for use in children's apps (as stated in their policies as of June 2020). Through analyzing traffic generated by the child's device, we confirmed that 11 Android solutions use prohibited SDKs.

PII Exposure to Third Parties. We found that all but one of the Android solutions share personal and unique device information with third-party domains. FamilyTime shares PII, including the child's name, email address, and phone number, with 11 third-party companies and the parent device's AAID with Facebook. The parent's phone number is shared with FastSpring.com, and the parent's email address is sent to 11 third parties. ScreenTime shares PII with four third-party companies, including the child's Android ID with Facebook.

COPPA Safe Harbor Providers. We checked the behavior of three of the 28 Android solutions (Kidoz, FamilyTime, and FindMyKids) certified by the U.S. FTC's COPPA Safe Harbor program (ftc.gov/safe-harbor-program). Our traffic analysis collected from the child's device reveals that FindMyKids uses three trackers and leaks AAID to at least two trackers: graph.facebook.com and adjust.com. FindMyKids sets two flags when calling Facebook to enable application tracking and advertiser tracking. FamilyTime sends the child's name, email address, and phone number (hashed in SHA256) to Facebook. Kidoz uses eight trackers and leaks the AAID to the third-party domain googleapis.com through the referer header.

Potential Practical Attacks

The impact of exploiting some of the discovered vulnerabilities in the analyzed parental control tools are summarized in this section.

Device Compromise

Device compromise presents serious security and privacy risks, especially if a vulnerability can be remotely exploited. We found multiple vulnerabilities in the Blocksi network device that can compromise the device itself. These include an exploitable command -injection vulnerability and a vulnerability in protecting the device's SN, which is used in authentication. A remote attacker can use these vulnerabilities to take control over the Blocksi device by simply knowing the parent's email address. In particular, using the SN and email address, an attacker can exploit the command-injection vulnerability and spawn a reverse TCP shell on the device. At this stage, the attacker gains full control of the device and can read/modify unencrypted network traffic and disrupt the router's operation (for example, Dynamic Host Configuration Protocol starvation)[9] or use it in a botnet (for example, Mirai).[10]

Account Takeover

Parental accounts can be compromised in multiple ways. First, none of the parental-control tools' web interfaces except Norton enforced HSTS, and most were found to be vulnerable to SSLStrip attacks. Therefore, an on-path attacker could possibly gain access to the parent's account using SSLStrip unless the parents carefully check the HTTPS status. Second, login pages that allow unlimited number of password trials could allow password guessing (especially for weak passwords). Note that most parental-control tools' password policies are apparently weak (for example, NIST).[11] Some products accept passwords as short as one character. Third, products with broken authentication allow access to parental accounts without login credentials. For example, SecureTeen provides an API endpoint for accessing the parental account by knowing only the parent's email address. If logged in, the attacker has access to a large amount of PII, social media/SMS messages, phone history, and the child's location—enabling possibilities of physical world attacks.

Data Leakage From Back Ends

Failure to protect the parental-control back-end databases exposes sensitive child/parent data at a large scale, which is exacerbated due to the collection and storage of a lot of user data by many solutions. Firebase misconfigurations expose data that belongs to more than 500 thousand children and parents from three apps. Such leakage may lead to potential exploitation of children, both online and offline.

Unprotected PII on the Network

Sending plaintext PII over the network is in direct violation of certain regulations, such as the U.S. COPPA, which mandates reasonable security procedures for protecting children's information.[12] We found that several parental-control tools transmit plaintext PII over the network, enabling any network attacker to have instant access to such sensitive data. For example, FindMyKids leaks surrounding voice and the child's picture, and MMGuardian leaks the child's geolocation. This could put a child in physical danger since the attacker can learn intimate details from the child's voice records and surroundings and identify the child from his/her photo or by using geolocation data. KidControl allows the child to send SOS messages when in a dangerous situation. However, an attacker can drop the SOS message at will as it is sent via HTTP. Moreover, KoalaSafe and Blocksi network devices append the child's device MAC address to outgoing DNS requests, enabling persistent tracking.

Recommendations for Solution Providers

- *Addressing vulnerabilities*: Because of the sensitivity of the information manipulated by parental-control tools, companies should conduct regular security audits. Our security and privacy framework can serve

as a starting point. Moreover, they should have a process to address vulnerabilities, such as responsible disclosure and bug bounty programs. Currently, no parental-control tools except Kaspersky and Bitdefender participate in such programs.

- *Enforcing best practices*: Parental-control companies should rely on publicly available guidelines and best practices, including proper API endpoint authentication and web-security standards, such as the Open Web Application Security Project recommendations. We also strongly encourage companies to adopt a strong-password policy in their products because the use of default, weak, and stolen credentials has been exploited in many known data breaches. In the case of network devices, manufacturers should employ a secure firmware-update architecture (for example, IETF).[13] Adopting known best practices is critical due to the especially vulnerable user base of these products.
- *Monitoring account activities*: Parental-control tools should report suspicious activities on the parent's account, such as password changes and accesses from unrecognized devices. These activities could indicate account compromise.
- *Limiting data collection:* Parental-control tools should limit the collection, storage, and transmission of the children's data to what is strictly necessary. For instance, the solution should not store PII not required for the solution's functionality. The parental-control tools should also allow the parent to selectively opt out of the data collection in certain features.
- *Securing communication:* Transmission of PII should happen exclusively over secure communication channels. The solution should utilize MITM mitigation techniques, such as host white-listing, certificate pinning, and HSTS.
- *Limiting third parties and SDKs:* Parental-control tools should avoid (or at least limit) the use of trackers and tracking SDKs in apps intended for children. They should use SDKs that are suitable for children (for example, Google has a list of third-party libraries that have been self-certified as compliant with child legislation). For the SDKs that allow special parameters for children's apps, those parameters must be used appropriately.

Our security and privacy evaluation identified several systematic problems in the design and deployment of most of the analyzed parental-control solutions across different platforms. Even though many parents may use these products as their children's digital guardians, several solutions can be abused to provide a new avenue to undermine children's online and real-world safety. Our findings call for greater scrutiny of these solutions, subjecting them to more rigorous and systematic evaluation and more stringent regulations. Parents should favor restriction apps over monitoring apps as monitoring apps generally collect more data and access more sensitive resources in a continuous manner. These data then become available to more third parties and perhaps even to attackers if the solution is not properly secured. Parents may also consider only using restrictions enabled by operating systems (available now in both desktop and mobile systems) to avoid exposure to third-party solution providers. A possibly better approach would be to use nontechnical measures, such as educating children about the safe and effective use of technology. ■

Acknowledgment

This work was partly supported by a grant from the Office of the Privacy Commissioner of Canada Contributions Program.

References

1. C. Anderson et al., "Are the kids alright? Digital risks to minors from South Korea's smart sheriff application," University of Toronto, Toronto, ON, Canada, 2015. [Online]. Available: https://citizenlab.ca/2015/09/digital-risks-south-korea-smart-sheriff/
2. Á. Feal, P. Calciati, N. Vallina-Rodriguez, C. Troncoso, and A. Gorla, "Angel or devil? a privacy study of mobile parental control apps," *Proc. Privacy Enhancing Technol.*, vol. 2020, no. 2, pp. 314–335, 2020. doi: 10.2478/popets-2020-0029.
3. I. Reyes et al., "Won't somebody think of the children?" examining COPPA compliance at scale," *Proc. Privacy Enhancing Technol.*, vol. 2018, no. 3, pp. 63–83, 2018. doi: 10.1515/popets-2018-0021.
4. B. Reaves et al., "Mo(bile) money, mo(bile) problems: Analysis of branchless banking applications," *ACM Trans. Privacy Security (TOPS)*, vol. 20, no. 3, pp. 1–31, 2017. doi: 10.1145/3092368.
5. X. de Carné de Carnavalet and M. Mannan, "Killed by proxy: Analyzing client-end TLS interception software," in *Proc. Netw. Distrib. Syst. Security Symp.*, pp. 1–17, 2016. doi: 10.14722/ndss.2016.23374.
6. S. Shasha, M. Mahmoud, M. Mannan, and A. Youssef, "Playing with danger: A taxonomy and evaluation of threats to smart toys," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2986–3002, 2019. doi: 10.1109/JIOT.2018.2877749.
7. A. Suzan, M. Elgharabawy, Q. Duchaussoy, M. Mannan, and A. Youssef, "Betrayed by the guardian: Security and privacy risks of parental control solutions," in *Proc. Annu. Comput. Security Appl. Conf.*, 2020, pp. 69–83. doi: 10.1145/3427228.3427287.
8. M. Cunche, "I know your MAC Address: Targeted tracking of individual using Wi-Fi," *J. Comput. Virology Hacking*

Techn., vol. 10, no. 4, pp. 219–227, 2014. doi: 10.1007/s11416-013-0196-1.

9. N. Tripathi, and N. Hubballi, "Exploiting DHCP server-side IP address conflict detection: A DHCP starvation attack," in *Proc. 2015 IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, pp. 1–3. doi: 10.1109/ANTS.2015.7413661.
10. M. Antonakakis et al., "Understanding the Mirai botnet," in *Proc. 26th USENIX security Symp. (USENIX Security 17)*, 2017, pp. 1093–1110.
11. P. A. Grassi et al., "Digital identity guidelines: Authentication and lifecycle management [includes updates as of 03-02-2020]," National Institute of Standards and Technology, Gaithersburg, MD, 2020. [Online]. Available: https://www.nist.gov/publications/digital-identity-guidelines-authentication-and-lifecycle-management-includes-updates-03
12. "Children's online privacy protection rule: A six-step compliance plan for your business," US Federal Trade Commission, Washington, D. C., 2017. [Online]. Available: https://www.ftc.gov/tips-advice/business-center/guidance/childrens-online-privacy-protection-rule-six-step-compliance
13. B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A firmware update architecture for internet of things. Internet-draft draft-ietf-suit-architecture-08," Internet Engineering Task Force, Wilmington, DE, 2019. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-suit-architecture-08.txt

Suzan Ali is a research assistant at the Madiba Security Research Lab, Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, H3G 1M8, Canada. Her research interests include security and privacy issues in public Wi-Fi hotspots and parental-control systems. Ali received a master's in information systems security from Concordia University. Contact her at suzanne_ali@hotmail.com.

Mounir Elgharabawy is a research assistant at the Madiba Security Research Lab, Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, H3G 1M8, Canada. His research interests include security and privacy issues in parental-control systems and IoT and Android firmware. Elgharabawy is currently pursuing a master's in information systems security at Concordia University. Contact him at mounir.elgharabawy@mail.concordia.ca.

Quentin Duchaussoy is a research assistant at the Madiba Security Research Lab, Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, H3G 1M8, Canada. His research interests include security and privacy issues in parental-control systems. Duchaussoy received a master's in information systems security from Concordia University. Contact him at duchaussoy@et.esiea.fr.

Mohammad Mannan is an associate professor at the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, H3G 1M8, Canada. His research interests include Internet and systems security with a focus on solving high-impact security and privacy problems of today's Internet. Mannan received a Ph.D. in computer science in the area of Internet authentication and usable security from Carleton University. Contact him at m.mannan@concordia.ca.

Amr Youssef is a professor at the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, H3G 1M8, Canada. His research interests include cryptography, network security, cyberphysical systems security, and privacy. Youssef received a Ph.D. in computer engineering in the area of cryptography from Queen's University, Ontario, Canada. Contact him at amr.youssef@concordia.ca.