Parental controls – Safer Internet solutions or new pitfalls?

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

***Abstract*—Parental control solutions are used by many parents to provide their children a safer digital environment. These 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.**

Index Terms: Privacy, Security, Mobile and Personal devices

THE INTRODUCTION

Many children are now as connected to the Internet as adults, 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 cyber-bullying and even cyber-crimes. To provide a safe internet experience, many parents rely on parental control solutions, which are also recommended by government agencies, including US FTC and UK 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, 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 privacy and security implications of parental control solutions, we design an experimental framework with a set of security and privacy tests, and systematically analyze popular representative solutions: 8 network devices, 8 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 among the solutions tested; the majority of solutions broadly fail to adequately preserve the security and privacy of their users—both children and parents.

Our notable findings include: (i) The Blocksi parental control router allows remote command injection, enabling an attacker with a parent's email address to eavesdrop/modify the home network's traffic, or use the device in a botnet (cf. Mirai). Blocksi's firmware update mechanism is also completely vulnerable to a network attacker. (ii) 9/28 Android solutions and 4/8 network devices do not properly authenticate their server

 Published by the IEEE Computer Society

API endpoints, allowing illegitimate access to view/modify server-stored children/parents data. (iii) 6/28 Android solutions allow an attacker to easily compromise the parent account at the server-end, enabling full account control to the child device (e.g., install/remove apps, allow/block phone calls and internet connections). (iv) 8/28 Android solutions transmit Personally Identifiable Information (PII) via HTTP (e.g., kidSAFE certified Kidoz sends account credentials via HTTP).

As part of responsible disclosure, we shared our findings and possible fixes with all solution providers. Two months after disclosure, only ten companies responded, with seven custom and three automatic replies. Notable changes after the disclosure: MMGuardian deprecated their custom browser; FamiSafe fixed the Firebase database security issue; and FamilyTime enabled HSTS on their server. Details of our findings and disclosure responses are available in the ACSAC version of our paper [7].

Related Work

Over the past years, several parental control tools have made the news for security and privacy breaches. Example exposures include: TeenSafe leaked thousands of children's Apple IDs and passwords; and Family Orbit exposed nearly 281 GB 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] of three popular Korean parenting apps mandated by the Korean government, revealing serious security and privacy issues in these apps. 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 (e.g., leakage of plaintext authentication information) using our comprehensive app analysis framework. Reyes et al. [3] analyzed children Android apps for COPPA compliance. Out of 5855 apps, the majority of the analyzed apps 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 the existing work and past security incidents, and 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 devices analyzed act as a man-in-the-middle between the client device and the internet router as follows: performing Address Resolution Protocol (ARP) spoofing, or creating a separate access point for all children's devices. ARP spoofing enables the network device to impersonate the home router, and monitor all local network traffic.

Android apps rely on several Android-specific mechanisms, including the following. (1) Device administration, which provides several administrative features at the system level, including: device lock, factory reset, certificate installation, and device storage encryption. (2) Mobile device management (MDM), which enables additional control and monitoring features, designed for businesses to fully control/deploy devices in an enterprise setting. (3) Android accessibility service, which enables capturing and retrieving window content, logging keystrokes, and controlling website content by injecting JavaScript code into visited web pages. (4) Android VPN, custom browsers, and third-party domain classifiers, which are used to filter web content. (5) Access to Facebook and YouTube OAuth credentials, which are used to monitor the 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 their usage and duration; and user activity is monitored via screenshots, keylogging, and accessing the webcam.

Parental control Chrome extensions use Chrome APIs to monitor the user-requested URLs, including: intercepting and redirecting traffic, modifying page content and meta-data

including cookies.

Threat Model

We consider the following attacker types with varying capabilities (but require no physical access to either the child/parent device or backend servers). (1) On-device attacker: a malicious app with limited permissions on the child/parent device. (2) Local network attacker: an attacker with direct or remote access to the same local network as the child device. (3) On-path attacker: a man-in-the-middle attacker between the home network and a solution's backend server. (4) Remote attacker: any attacker who can connect to a solution's backend 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], [5], [6], and then iteratively refined by us.

1) *Vulnerable client product*: A parental control product (including its update mechanism) being vulnerable, allowing sensitive information disclosure (e.g., via on-device side-channels), or even full product compromise (e.g., via arbitrary code execution).
2) *Vulnerable backend*: The use of remotely exploitable outdated server software, and misconfigured or unauthenticated backend API endpoints (e.g., Google Firebase in Android apps).
3) *Improper access control*: Failure to properly check whether the requester owns the account before accepting queries at the server-end (e.g., insecure direct object reference).
4) *Insecure authentication secrets*: Plaintext storage or transmission of authentication secrets (e.g., passwords and session IDs).
5) *SSLStrip attack*: The parental control tool's online management interface is vulnerable to SSLStrip attacks that strip away the security provided by HTTPS, exposing private information in plaintext; countermeasures exist (e.g., HSTS) but must be correctly implemented.
6) *Weak password policy*: Acceptance of very weak passwords (e.g., with 4 characters or less).
7) *Online password brute-force*: No defense against unlimited login attempts on the online parental login interface (e.g., CAPTCHA).
8) *Uninformed suspicious activities*: No notifications to parents about indicators of possible compromise (e.g., the use of parental accounts on a new device, or password changes).
9) *Insecure PII transmission*: PII from the client-end is sent without encryption, allowing an adversary to eavesdrop for PII.
10) *PII exposure to third-parties*: 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 over 10K+ installations from Google Play, four companion apps for network devices, six additional apps available on their official websites with additional features, making the total to 153 (after removing 15 unresponsive/unrelated apps); 51/153 are pure children apps; 24 are pure parent apps; and 78 are used for both parent and child devices. For in-depth analysis, we picked 46 popular Android apps representing 28 parental control solutions.

Methodology

We combine dynamic (primarily traffic and usage) and static (primarily code review/reverse-engineering) analysis to identify security and privacy flaws in parental control tools; for an overview, see Fig. 1. For each product, we first conduct a dynamic analysis and capture the parental control tool traffic during its usage

(as parents/children); if the traffic is in plaintext or decryptable (e.g., via TLS interception), we also analyze the information sent. Second, we statically analyze their binaries (via reverse engineering) and scripts (if available). We pay specific attention to the API requests and URLs present in the code to complement the dynamic analysis. After merging the findings, we look into the domains contacted and check the traffic for security flaws (e.g., TLS weaknesses). Third, we test the security and privacy issues listed under "Potential Security and Privacy Issues" against the collected API URLs and requests. For the parental control tool with an online interface, we assess the password-related issues and test the SSLStrip attack against the login page.

Dynamic Analysis

Usage Emulation and Experimental Setup. We set up test environments for each solution, emulate user actions for hours to days, with the goal of triggering UI events looking for signs of PII leakage, weak security measures, or potential vulnerabilities, and collect the traffic from the child, parent, and network devices, and then perform relevant analysis. We evaluate the web filtering mechanism by visiting a blocked website (gambling/adult) and a university website. We also perform user activities monitored by platform-specific parental control features and evaluate the solution's operations. For example, on Android, we perform basic phone activities (SMS, phone call) and internet activities (Instant messaging, social media, browsing, and accessing blocked content).

We evaluate 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 use test devices with web browsing to emulate a child's device. If the parental control device uses ARP spoofing, the test device is connected directly to the router's wireless access point (AP); otherwise, the test device is connected to the parental control device's wireless AP. We capture network traffic on both the test device and the router using Wireshark and tcpdump, respectively.

For Android apps, we use separate Android phones to concurrently record and inspect network traffic originating from the child and parent apps. We test each Windows application and Chrome extension on a fresh Windows 10 virtual machine with Chrome, and mitmproxy installed.
Traffic Analysis. After intercepting traffic, we parse and commit the collected traffic to an SQLite database and check for the following security and privacy related issues.

We check for PII and authentication secrets transmitted in plaintext, or leakage of PII to third-party domains. We automatically search for PII items (i.e., case-insensitive partial string match) in the collected traffic, and record the leaked information, including the HTTP request URL. We decode the collected network traffic using common encoding (base64 and URL encoding) and encode 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 identify all the APIs that can be potentially exploited (without strong authentication), by replaying the recorded HTTP request stripped of authentication headers (e.g., cookies and authorization header). Then, we retrieve the parameters used by these APIs (e.g., keys, tokens, or unique IDs), and assess the parameters in terms of their predictability and confidentiality.

We compile lists of known trackers, and then identify communication to these trackers and other third-party SDKs in the parental control tools traffic (third-parties are defined as any domain other than the product providers).
Backend Assessment. We only look into the backends' software components as disclosed by web servers or frameworks in their HTTP response headers, such as "Server" and "X-Powered-By". We then match these components against the CVE database to detect known vulnerabilities associated with these versions.

Static Analysis

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

We analyze the network device firmware whenever possible. We either attempt to extract

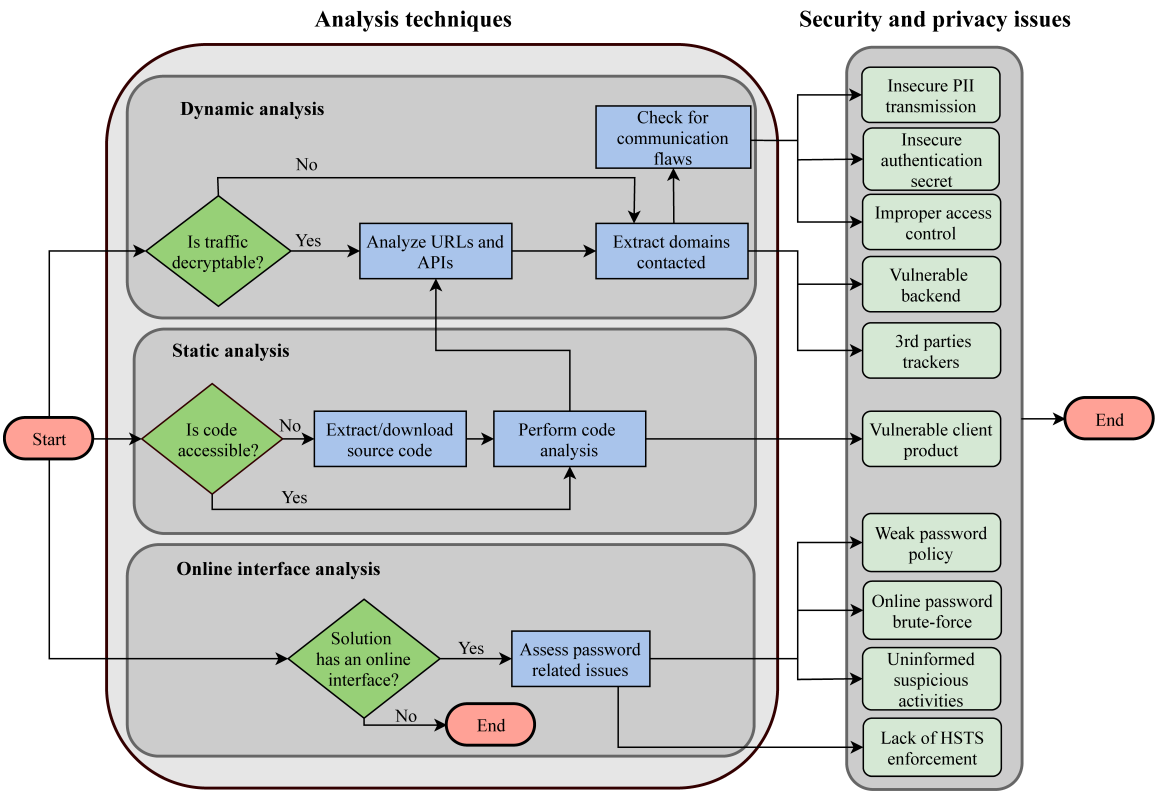

Figure 1. Overview of our evaluation framework.

the firmware directly from the device (via physical interfaces), or download the device firmware from the vendor's website. We then scan the network devices with several tools (OpenVas, Nmap, Nikto and Routersploit), and match the identified software versions against public vulnerability databases.

We manually analyze the source code of the Chrome extensions, which mainly consists of scripts, separated into content scripts and background scripts. We perform an automated analysis on all 153 Android apps using the Firebase Scanner (github.com/shivsahni/FireBaseScanner) to detect security misconfigurations in Firebase, a widely used backend infrastructure management for Android apps. We also use 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 use Exodus-Privacy (reports.exodus-privacy.eu.org/en/trackers/) to classify tracking SDKs. We use MOBSF (github.com/MobSF/Mobile-Security-Framework-MobSF) to extract the list of third-party tracking SDKs from all 153 apps based on Exodus-Privacy's tracker list.

Online Interface Analysis

The online user interface 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 the parent account can be very damaging, and thus we evaluate the security of this interface. To check for SSLStrip attacks, we first set up a WiFi Access Point with a set of network interception tools installed. Then, we connect the parental control tool to our WiFi AP, and launch the SSLStrip script (github.com/moxie0/sslstrip). We confirm the effectiveness of the attack by comparing the result to the corresponding traffic in a regular testing environment. To test the password policy, we check if the service accepts a password with 4 characters or less. We also use Burp Suite (portswigger.net/burp) to perform password

brute-force attacks, and limit our script only to 50 authentication attempts on our own account from a single computer. We also test two scenarios in which a parent should be notified (e.g., via email): modification of the user's password, and connection to the account from a new/unknown device.

Results

We analyzed the parental control tools between Mar. 2019 to Sep. 2020, which include: 8 network devices, 46 Android apps representing 28 Android solutions, 10 Chrome extensions and 8 Windows applications. We present some of the most prominent findings on the tested security and privacy issues (further details in [7]); for an overview see Table 1.

Vulnerable Client Product

Network devices. The Blocksi firmware update happens fully through HTTP. An integrity check is done on the downloaded binary image, using an unkeyed SHA256 hash, again retrieved using HTTP, and thus rendering it useless. Therefore, an on-path attacker can trivially alter the update file and inject their own malicious firmware into the device; see Fig. 2.

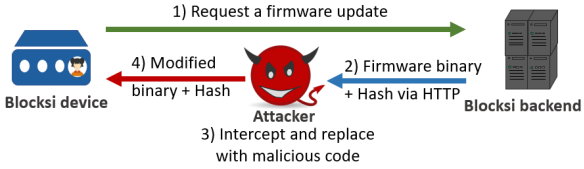

Figure 2. Blocksi update mechanism flaw.

Android apps. We found 3/28 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 SD card. Examples of the sensitive information include: the parent's email and PIN code, phone numbers, the child's geolocation data, messages and social media chats, visited websites, and even authentication tokens—which enabled us to read private information from the child account remotely. In addition, Kidoz, KidsPlace, and MMGuardian use custom browsers to restrict and filter web content. The three browsers fail to enforce HSTS (a security protocol design to protect against SSLStrip attack), and lack persistent visual indication if the website is served on HTTP.

Windows applications and Chrome extensions. Other than Kidswatch, all 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 run time, bypassing the static control of Chrome for extension security, which has been exploited in the wild by tricking developers into adding malicious scripts masquerading as tracking scripts.

Vulnerable Backend

Android apps. Google Firebase is a popular backend service used in 115/153 of our Android apps dataset. Critical misconfigurations can allow attackers to retrieve all unprotected data stored on the cloud server. We found 8 Android apps with insecure Firebase configurations; prominent exposures include (verified using our own accounts): FamiSafe with 500K+ installs exposes the parent email; Locate with 10K+ installs exposes the child name, phone number, and email; and My Family Online with 10K+ installs exposes the child name, child and parent phone numbers, parent email, and apps installed on child phone. Following our disclosure, FamiSafe fixed the Firebase security issue.

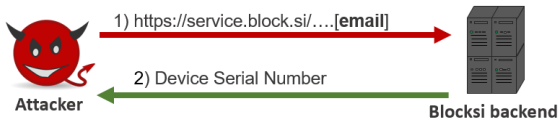

Figure 3. Blocksi improper access control.

Improper Access Control

Network devices. For Blocksi's login API endpoint, the device's serial number (SN) and the registered user's email 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

Table 1. Most significant results for security flaws in parental control tools labelled following our threat model. ○ : 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 2 types of attackers, we display the fullest circle applicable.

	Network devices						Android solutions																Chrome			Windows			
Security Flaw	**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**	**Saferkid**	**Blocksi Web Filter**	**Porn Blocker**	**FamilyFriendly**	**Qustodio**	**Dr. Web**	**Spyrix**	**KidLogger**
Vulnerable client product		●	◔	◐				○	◐		◐	○	◐								○								◐	◐	
Vulnerable backend	●	●	●		●	●		●					◐	◐			◐											◐			◐
Improper access control	◔	◔		●	◐		●	○		◐	◐		◐			●	●						●		–	–	–				
Insecure authentication secret			◔						◐		◐		◐		◐		●						●		–	–	–				
SSLStrip attack	–	◐	◐	◐			◐	◐	◐	◐	◐	◐		◐	◐		◐	–	◐	◐	–	–	–	–	–	–	–	◐	–	◐	◐
Online password bruteforce			●	●			●	●	●	●		●	●	●	●	●	●	●	●	●	–	●	●	●	–	–	–	●	–	●	●
Weak password policy		●		●			●			●			●	●			●	●	●		–				–	–	–		–	●	●
Uninformed suspicious activities	●	●		●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	–			●	–	–	–	●	–	●	●
Insecure PII transmission	◐	◐		◐	◐					◐	◐	◐	◐	◐			◐				○				◐	◐	◐				
PII exposure to third-parties	●						●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●				

user's email using their device SN or vice-versa, and thus access sensitive information about the home network, e.g., the WiFi password, and MAC addresses of connected devices, see Fig. 3.

To authenticate to HomeHalo's API endpoint, it uses only the device's SN and an HTTP header called `secretToken` (an apparently fixed value of 100500). An on-path attacker can intercept and modify these messages, and gain access to admin controls, e.g., the wireless SSID, password, or even the device's root password.

Android apps. We found 9/28 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, allowing the attacker to monitor and control the child 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 name, gender, date of birth, email address, and child phone number (by simply trying all 6-digit values for childID). In FamiSafe, an attacker app can retrieve all the child 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 installed apps). Bosco's API endpoint fails to check the relation between the provided secure authorization token provided and the information requested, allowing any parent account with a valid token to request information about any registered child knowing only its ID, which is set to the Android advertising ID (AAID) by Bosco. However, AAID is available to all apps, and thus, enabling an attacker with an app on the child device to easily retrieve child PII (e.g., child geolocation, phone call history, and pictures).

Insecure Authentication Secret

Network devices. 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. Consequently, as this AP is open and the client-device communication happens through HTTP, the home router's WAN and KidsWifi's LAN credentials become available to local attackers.

Android apps. Kidoz exposes the user email 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, history of movements,

and remote control functions on the child phone (e.g., block all phone calls) in Qustodio. For KidsPlace, the attacker can lock the child phone, disable the Internet, install malicious apps and upload harmful content to the child mobile.

SSLStrip and Online Account Issues

We found that 11 Android solutions, four network devices and three Windows applications transmitted the parent 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 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 (i.e., 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 serial number into outgoing 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 backend server. Several Android solutions also send cleartext PII, including: FindMyKids (the child's surrounding sounds and photo); KidControl (the parent's name and email, geolocation, and SOS requests); and MMGuardian (the parent's email and phone number, and child's geolocation).

Third-party SDKs and Trackers

Some legislations (e.g., US COPPA and EU GDPR) regulate the use of third-party trackers in the services targeting children (e.g., under 13 years of age). We thus evaluate 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 identify several tracking third-party SDKs from network traffic generated during our dynamic analysis from child device. Except SecureTeen and Easy parental control, 26/28 Android solutions use tracking SDKs, 1–16 unique trackers. Our traffic analysis confirms violations of COPPA—over 30% of Android solutions utilize doubleclick.net without passing the proper COPPA compliant parameter from child device. We also found that one of the network devices' companion app, Circle, includes a third-party analytical SDK from Kochava, and shares the following: Device ID (enables tracking across apps), 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 study the SDKs identified in past studies [3], [2] that are restricted by their developers (e.g., fully prohibited, or use with particular parameters) for use in children's apps (as stated in their policies as of June 2020). Through analysing traffic generated by the child device, we confirm that 11 Android solutions use prohibited SDKs.

PII exposure to third-parties. We found that all but one Android solution share personal and unique device information with third-party domains. Prominent examples include the following. FamilyTime shares PII with 11 third-party companies, including, the child name, email and phone number, and parent device's Android Advertising ID (AAID) with Facebook; the parent phone number is shared with FastSpring.com, and the parent email is sent to 11 third-parties. ScreenTime shares PII with four third-party companies, including the child Android ID with Facebook.

COPPA Safe Harbor providers. We check the behavior of (3/28) (Kidoz, FamilyTime, FindMyKids) Android solutions certified by the US FTC's COPPA Safe Harbor program (ftc.gov/safe-harbor-program). Our traffic analysis collected from the child 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

Device compromise. Device compromise presents serious security and privacy risks, especially if a vulnerability can be exploited remotely. 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 serial number, 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 serial number and email, 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, disrupt the router's operation (cf. DHCP starvation [9]), or use it in a botnet (cf. Mirai [10]).

Account takeover. Parental accounts can be compromised in multiple ways. First, none of the parental control tools' web interface except Norton enforced HSTS, and most were found vulnerable to SSLStrip attacks. Therefore, an on-path attacker can possibly gain access to the parent account using SSLStrip, unless 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 (cf. 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 to access the parental account, by knowing only the parent email address. If logged-in, the attacker has access to a large amount of PII, social media/SMS messages, phone history, child location—even enabling possibilities of physical world attacks.

Data leakage from backends. Failure to protect the parental control backend databases exposes sensitive child/parent data at a large scale, exacerbated due to the collection and storage of a lot user data by many solutions. Firebase misconfigurations exposed data that belongs to 500K+ 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 US COPPA, which mandates reasonable security procedures for protecting children's information [12]. We found several parental control tools transmit plaintext PII over the network, enabling any network attacker 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 her surrounding, and also identify the child from her photo, or by using geolocation data. KidControl allows the child to send SOS messages when she is 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 the 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, none 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 stan-

dards (e.g., OWASP 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 (see e.g., 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 whitelisting, 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 (e.g., Google has a list of third party libraries that have been self-certified as compliant with children legislation). For the SDKs that allow special parameter for children's apps, those parameters must be used appropriately.

Conclusion

Our security and privacy evaluation identified several systematic problems in the design and deployment of most analyzed parental control solutions across different platforms. Even though many parents may use these products as their children's digital guardians, several solutions in fact 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. For parents, they should favor restriction apps over monitoring apps, as monitoring apps generally collect more data and access more sensitive resources in a continuous manner, which then become available to more third-parties, and perhaps even to attackers if the solution is not properly secured. Parents may also consider using only 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 non-technical measures such as educating children about safe and effective use of technology.

Acknowledgment

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

REFERENCES

1. C. Anderson, M. Crete-Nishihata, C. Dehghanpoor, R. Deibert, S. McKune, D. Ottenheimer, J. Scott-Railton, "Are the Kids Alright? Digital Risks to Minors from South Korea's Smart Sheriff Application," 2015.
2. Á. Feal, P. Calciati, N. Vallina-Rodriguez, C. Troncoso, A. Gorla, "Angel or devil? a privacy study of mobile parental control apps," Proceedings on Privacy Enhancing Technologies 2020, no. 2 (2020): 314-335.
3. I. Reyes, P. Wijesekera, J. Reardon, A. Elazari Bar On, A. Razaghpanah, N. Vallina-Rodriguez, S. Egelman, ""Won't somebody think of the children?" examining COPPA compliance at scale," Proceedings on Privacy Enhancing Technologies 2018, no. 3 (2018): 63-83.
4. B. Reaves, J. Bowers, N. Scaife, A. Bates, A. Bhartiya, P. Traynor, K.R. Butler, "Mo(bile) money, mo(bile) problems: Analysis of branchless banking applications," ACM Transactions on Privacy and Security (TOPS) 20, no. 3 (2017): 1-31.
5. X. de Carné de Carnavalet, M. Mannan, "Killed by proxy: Analyzing client-end TLS interception software," In Network and Distributed System Security Symposium. 2016.
6. S. Shasha, M. Mahmoud, M. Mannan, A. Youssef, "Playing with danger: A taxonomy and evaluation of threats to smart toys," IEEE Internet of Things Journal 6, no. 2 (2019): 2986-3002.
7. A. Suzan, M. Elgharabawy, Q. Duchaussoy, M. Mannan, A. Youssef, "Betrayed by the Guardian: Security and Privacy Risks of Parental Control Solutions," In Annual

Computer Security Applications Conference, pp. 69-83. 2020.

8. M. Cunche, "I know your MAC Address: Targeted tracking of individual using Wi-Fi," Journal of Computer Virology and Hacking Techniques 10, no. 4 (2014): 219-227.
9. N. Tripathi, and N. Hubballi, "Exploiting dhcp server-side ip address conflict detection: A dhcp starvation attack," In 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), pp. 1-3. IEEE, 2015.
10. M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric et al, "Understanding the mirai botnet," In 26th USENIX security symposium (USENIX Security 17), pp. 1093-1110. 2017.
11. P. A. Grassi, J. L. Fenton, E. M. Newton, R. A. Perlner, A. R. Regenscheid, W. E. Burr, J. P. Richer et al, "Digital Identity Guidelines: Authentication and Lifecycle Management [includes updates as of 03-02-2020]," (2020).
12. US Federal Trade Commission, "Children's online privacy protection rule: A six-step compliance plan for your business," https://www.ftc.gov/tips-advice/business-center/guidance/childrens-online-privacy-protection-rule-six-step-compliance. 2017.
13. B. Moran, H. Tschofenig, D. Brown, M. Meriac, "A Firmware Update Architecture for Internet of Things. Internet-Draft draft-ietf-suit-architecture-08. Internet Engineering Task Force," https://datatracker.ietf.org/doc/html/draft-ietf-suit-architecture-08. Work in Progress. 2019.

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. She received her Master's degree in Information Systems Security from Concordia University in 2020. Her research interests include security and privacy issues in public WiFi hotspots and parental control systems. 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. He is currently pursuing his Master's degree in Information Systems Security at Concordia University. His research interests include security and privacy issues in parental control systems, and IoT and Android firmware. 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. He received his Master's degree in Information Systems Security from Concordia University in 2020. His research interests include security and privacy issues in parental control systems. 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. He received a Ph.D. in computer science in the area of Internet authentication and usable security from Carleton University. His research interests lie in the area of Internet and systems security, with a focus on solving high-impact security and privacy problems of today's Internet. 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. He received a Ph.D. in computer engineering in the area of cryptography from Queen's University, Ontario, Canada. His research interests include cryptography, network security, cyber-physical systems security, and privacy. Contact him at amr.youssef@concordia.ca.