Betrayed by the Guardian: Security and Privacy Risks of Parental Control Solutions

Suzan Ali
a_suzan@ciise.concordia.ca
Concordia University
Montreal, Quebec, Canada

Mounir Elgharabawy
m_elghar@encs.concordia.ca
Concordia University
Montreal, Quebec, Canada

Quentin Duchaussoy
q_duchau@encs.concordia.ca
Concordia University
Montreal, Quebec, Canada

Mohammad Mannan
m.mannan@concordia.ca
Concordia University
Montreal, Quebec, Canada

Amr Youssef
youssef@ciise.concordia.ca
Concordia University
Montreal, Quebec, Canada

ABSTRACT

For parents of young children and adolescents, the digital age has introduced many new challenges, including excessive screen time, inappropriate online content, cyber predators, and cyberbullying. To address these challenges, many parents rely on numerous parental control solutions on different platforms, including parental control network devices (e.g., WiFi routers) and software applications on mobile devices and laptops. While these parental control solutions may help digital parenting, they may also introduce serious security and privacy risks to children and parents, due to their elevated privileges and having access to a significant amount of privacy-sensitive data. In this paper, we present an experimental framework for systematically evaluating security and privacy issues in parental control software and hardware solutions. Using the developed framework, we provide the first comprehensive study of parental control tools on multiple platforms including network devices, Windows applications, Chrome extensions and Android apps. Our analysis uncovers pervasive security and privacy issues that can lead to leakage of private information, and/or allow an adversary to fully control the parental control solution, and thereby may directly aid cyberbullying and cyber predators.

CCS CONCEPTS

• **Security and privacy → Systems security**.

KEYWORDS

Parental control network devices, Android apps, Windows applications, Web extensions, Privacy, Security

ACM Reference Format:
Suzan Ali, Mounir Elgharabawy, Quentin Duchaussoy, Mohammad Mannan, and Amr Youssef. 2020. Betrayed by the Guardian: Security and Privacy Risks of Parental Control Solutions. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3427228.3427287

ACSAC 2020, December 7–11, 2020, Austin, USA

ACM ISBN 978-1-4503-8858-0/20/12…$15.00
https://doi.org/10.1145/3427228.3427287

1 INTRODUCTION

Many of today's children cannot imagine their daily lives without internet. A recent survey [66] shows that 42% of US children (4–14 years) spend over 30 hours a week on their phones; nearly 70% of parents think that such use has a positive effect on their children's development [66]. While the web could be an excellent environment for learning and socializing, there is also a plethora of online content that can be seriously damaging to children. In addition, children are by nature vulnerable to online exploitation and other risk effects of online social networking, including cyber-bullying and even cyber-crimes (see e.g., [3, 20]); the current COVID-19 pandemic has only increased these risks (see e.g., [73]).

To provide a safe, controlled internet experience, many parents and school administrators rely on parental control solutions that are easily accessible either for free, or for a relatively cheap price. From recent surveys in the US, some forms of parental control apps/services are used by 26–39% of parents [17, 53], indicating a growing adoption of these solutions. Such solutions are also recommended by government agencies, e.g., US FTC [31] and UK Council for Child Internet Safety (UKCCIS) [72], despite their limited effectiveness (cf. EU commissioned benchmark at: sipbench.eu), and questionable morality since they, arguably, can act as surveillance tools [79]. This ethical/moral debate is outside the scope of our work.

On the other hand, over the past few years, many attacks targeted parental control solutions, exposing monitored children's data, sometimes at a large scale [46, 54]. Aside from endangering children's safety (online and in the real-world), such leaked children's personal data may be sold by criminals (cf. [81]). Recent reports also revealed several security and privacy issues in the analyzed parental control solutions [4, 28, 77]. However, such analysis was limited to the privacy of Android apps, and only one network device, even though popular parental control solutions are used across different platforms: mobile and desktop OSes, web extensions, and network devices. Note that, unlike other vulnerable products (e.g., buggy gaming apps [78]), or non-complaint products (e.g., Android apps for children [62]), which can be removed when such concerns are known, parental control solutions are deemed

essential by many parents and schools, and thus are not expected to be removed due to the lack of better alternatives.

We undertake the first comprehensive study to analyze different types of parental control hardware and software solutions. We design a set of security and privacy tests, and systematically analyze popular representative parental control solutions available in network devices, Windows and Android OSes, and Chrome extensions. While developing our comprehensive analysis framework for solutions in multiple platforms, we faced several challenges. Most parental control solutions implement various techniques that hinder traffic analysis (e.g., VPNs, SSLpinning and custom protocols). The use of proprietary firmware and code obfuscation techniques also poses challenges for static analysis. Understanding long-term behaviors of these solutions by running them for hours/days in a realistic way (e.g., triggering all their features), is also time consuming (compared to simple, automated UI fuzzing).

Contributions. Our contributions can be summarized as follows. (i) We developed an experimental framework for systematically evaluating security and privacy issues in parental control software and hardware solutions. (ii) We utilized this framework to conduct the first comprehensive study of parental control tools on multiple platforms, including 8 network devices, 8 Windows applications, 10 Chrome extensions, and 29 Android apps representing 13 Android solutions grouped by vendor.[1] The in-depth analysis aims to inspect the apps' web traffic for personally identifiable information (PII) leakage, insecure API endpoints authentication, potential vulnerabilities, and the presence of third-parties and known trackers. (iii) Our analysis reveals 135 vulnerabilities among the solutions tested and highlights that the majority of solutions broadly fail to adequately preserve the security and privacy of their users—both children and parents.

Notable findings and disclosure. Our notable findings include:

- The Blocksi parental control router allows remote command injections, 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 [8]). Blocksi's firmware update mechanism is also completely vulnerable to a network attacker.
- 8/13 Android solutions and 4/8 network devices do not properly authenticate their server API endpoints, allowing illegitimate access to view/modify server-stored children/parents data.
- 5/13 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).
- 7/13 Android solutions transmit PII via HTTP (e.g., kidSAFE [64] certified Kidoz sends account credentials via HTTP).
- Among the parental control tools with a web interface, 9/13 Android solutions, 4/8 network devices, and 3/8 Windows applications are vulnerable to SSLStrip attacks (cf. [43, 44]), a man-in-the-middle (MITM) attack, due to the lack of HSTS.
- 2/8 Windows applications utilize a TLS proxy that degrades connection security, by accepting certificates and ciphers that are rejected by modern browsers. Another Windows application (Kidswatch) completely lacks HTTPS, and communicates with the backend server via HTTP.
- 2/10 Chrome extensions and 4/13 Android solutions transmit the full URLs from the browser to their server, possibly leaking sensitive (session) information.

As part of responsible disclosure, we contacted the developers of the solutions we analyzed, and shared our findings, including proof-of-concept scenarios and possible fixes. Two months after disclosure, only ten companies responded, seven custom and three automatic replies. Blocksi, KoalaSafe, MMGuardian, KidsPlace, FamiSafe, and FamilyTime responded that they are investigating the issues. Kidoz responded that some of our reported issues are on their fixing backlog, and acknowledged the new vulnerabilities. Other vendors either sent automatic/ambiguous response, or no response at all. Notable changes after the disclosure: MMGuardian deprecated their custom browser; FamiSafe fixed the Firebase database security issue; and FamilyTime enabled HSTS on their server.

2 RELATED WORK

In this section, we first list a few example cases from real-world data breaches involving parental control tools, and then summarize related academic studies (mostly privacy analyses of Android apps).

Over the past years, several parental control tools have made the news for security and privacy breaches. The teen-monitoring app TeenSafe leaked thousands of children's Apple IDs, email addresses and passwords [46]. Family Orbit exposed nearly 281 GB of children data from an unsecured cloud server [54]. In 2019, a privacy flaw in Kaspersky anti-virus and parental control application was found [24]. This application included a script to perform content checking on each page intercepted by a TLS proxy. However, some unique IDs were also included in the process, allowing the website to track the user. In 2010, EchoMetrix settled US FTC charges for collecting and selling children's information to third-parties through their parental control software [25].

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 of three popular Korean parenting apps mandated by the Korean government, Smart Sheriff, Cyber Security Zone and Smart Dream [4]. The security audits found serious security and privacy issues in the three parental control Android apps. For example, Smart Sheriff failed to adequately encrypt PII either on storage or in transit. Smart Dream allowed unauthorized access to children's messages and search history.

Feal et al. [28] studied 46 parental control Android apps for data collection and data sharing practices, and the completeness and correctness of their privacy policies. They used the Lumen Android app (see https://haystack.mobi/) for their analysis, which is unable to analyze target apps with VPN or certificate pinning. Parental apps and dashboards are also excluded. Our analysis framework has no such limitations, and consequently we are able to identify new critical security issues (e.g., leakage of plaintext authentication information), even among the apps analyzed by Feal et al.

Reyes et al. [62] 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. Wisniewski et al. [79] evaluated 42 features

[1] An Android solution is typically composed of a child app, a parent app, and an online parental dashboard. We consider an Android solution vulnerable if any of its component is vulnerable.

in 75 parental control Android apps, showing that most apps value control over self-regulation strategies, and boast the use of privacy invasive techniques. Marsh [47] measured the effectiveness and usability of two parental control apps.

Web extensions have been subjected to security evaluation for over a decade (see e.g., [14, 69]), but no past studies focused on parental control extensions. Windows parental control applications have been only studied for the security of their TLS proxies [19]. Similarly, parental control network devices remained unexplored, except the Disney Circle, analyzed by Cisco Talos in 2017, and found to have 23 different security vulnerabilities [77]. Among other devices, we also analyzed Circle, but used a newer version released in 2019.

In contrast to previous work, we conduct a comprehensive, systematic study of security and privacy threats in parental control solutions across multiple platforms: mobile (Android), desktop (Windows), web browser (Chrome extensions) and stand-alone network devices, as popular solutions are available in all these platforms. Our analysis therefore sheds light on the broader picture of security and privacy risks of parental control tools. Compared to existing Android app studies, our framework is more in-depth (e.g., monitoring the apps from the OS instead of the application level), and inclusive (e.g., analyze apps with VPNs and key pinning).

3 BACKGROUND AND THREAT MODEL

We use the term "parental control tools" to cover different types of parental solutions: network devices, Android apps, Chrome extensions and Windows applications. Personally identifiable information (PII) refers to any information related to the user as defined by the US FTC and Office of the Privacy Commissioner of Canada. Any entity that is not directly related to a parental control solution, is labelled as a third-party; this includes but is not limited to trackers and advertisers. In what follows, we briefly discuss some common techniques used by parental control tools, define our threat model, and list the vulnerabilities that we test against each solution.

3.1 Monitoring Techniques

Parental control tools generally allow the parent to remotely control the child device, perform web filtering, and monitor activities on social media. We derive the following monitoring techniques from product documentation, our observations from installation procedure and use/analysis of these solutions. These techniques vary significantly across platforms, and are grouped here as such.

Network devices. Being network-based, parental control devices can monitor network traffic but 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 by using one of two techniques: performing Address Resolution Protocol (ARP) spoofing, or creating a separate access point. ARP spoofing enables the network device to impersonate the internet router on the local network. The device achieves that by sending forged ARP packets that bind the router's IP with the network device's MAC address. As a result, all the local network traffic is routed through the device before going to the internet router. Alternatively, the network device may create an explicit access point exclusively for children to enforce parental control filtering on all devices connected to it.

Android apps. Android apps rely on several Android-specific mechanisms, including the following (see Table 6 in Appendix for per Android solution capabilities). (1) Device administration [5, 67] provides several administrative features at the system level, including: device lock, factory reset, certificate installation, and device storage encryption. (2) Mobile device management (MDM [45]) enables additional control and monitoring features, designed for businesses to fully control/deploy devices in an enterprise setting.[2] (3) Android accessibility service [6, 67] enables apps to perform several functions including monitoring user actions by receiving notifications when the user interacts with an app, capturing and retrieving window content, logging keystrokes, and controlling website content by injecting JavaScript code into visited web pages. (4) Notification access enables Android apps to read or dismiss all notifications displayed in the status bar; notifications may include personal information such as contact names and messages. (5) Android VPN, custom browsers, and third-party domain classifiers (e.g., Komodia.com [39]), which are used to filter web content. (6) Facebook [27] and YouTube OAuth [33] features, which are used to monitor the child's activities on Facebook (e.g., posts and photos), and YouTube (e.g., playlists and comments). (7) Miscellaneous techniques including: having browser history and bookmarks permission, using custom browsers, or TLS interceptions via Android VPN.

Windows applications. As opposed to Android parental control apps, Windows applications operate with more privileges, and use the following techniques: (1) TLS-interception: a proxy is installed by inserting a self-signed certificate in the trusted root certificate store. This allows the Windows applications to perform content analysis and alter content from HTTPS webpages. (2) Application monitoring: user applications are monitored for their usage and duration. (3) User activity monitoring: some Windows applications take screenshots, record keystrokes, and access the webcam.

Chrome extensions. With appropriate permissions, a parental control extension can use the Chrome API and retrieve the URL contacted by the user, intercept and redirect traffic, read and modify page content and meta-data including cookies.

3.2 Threat Model

We consider the following attacker types with varying capabilities. (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. This attacker can eavesdrop, modify, and drop messages from the local network. (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. Attacks requiring physical access to either the child/parent device are excluded from our threat model.

[2] Note that, MDM features may be just too powerful, and may enable dangerous remote control operations including device wipe. Apple has removed several popular parental control apps from App Store due to their use of such highly invasive features (https://www.apple.com/ca/newsroom/2019/04/the-facts-about-parental-control-apps/). In contrast, Google Play apparently still allows these features in parental apps.

3.3 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 is initially inspired by previous work [4, 19, 60, 68], 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 [35] 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 attack, possibly due to lack of HSTS enforcement (cf. [43, 44]) .
(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.
(8) *Uninformed suspicious activities*: No notifications to parents about potentially dangerous activities (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.

3.4 Selection of Parental Control Solutions

We chose solutions used in the most popular computing platforms for mobile devices (Android), personal computers (Windows), web browser (Chrome), and selected network products from popular online marketplaces (Amazon).[3] 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 (e.g., [13, 38, 52]), and selected eight applications. For Android apps, we searched the following terms on Google Play: "Parental Control" and "Family Tracker". From a total of 462 apps, we selected 158 apps with over 10K+ installations, and analyzed them automatically. We also downloaded the companion apps for four network devices (Circle companion app was already in our dataset as it had 50K+ installs). For six of these apps, the developers made available (via their official websites) alternative APKs with additional features. These APKs were also included in the set of automatically analysed apps, adding up to 168 apps. We installed these apps on an Android phone and removed 15 unresponsive/unrelated apps, making the total of apps analyzed to 153; 51/153 are pure children apps; 24 are pure parent apps; and 78 are used for both parent and child devices, which we termed as "shared apps". For in-depth analysis, we picked 29 popular Android apps representing 13 parental control solutions. Each solution may include child app(s), parent app(s), and online parental dashboard.

[3] As of May 2020, current market shares according one estimate (https://gs.statcounter.com) are: Android 72.6%, Windows 77% and Chrome 63.9%.

4 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 MITM), 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 described in Sec. 3.3 against the collected API URLs and requests. Lastly, in case the parental control tool presents an online interface, we assess the password-related issues and test the SSLStrip attack against the login page.

4.1 Dynamic Analysis

We set up test environments for each solution, emulate user actions for hours to days, collect the traffic from the child, parent, and network devices, and then perform relevant analysis (see Sec. 3.3).

4.1.1 Usage Emulation and Experimental Setup. We analyze each solution by manually mimicking regular users' operations with the goal of triggering parental control mechanisms. We test for potential vulnerabilities in these mechanisms (see Sec. 4.1.2). 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 (see Sec. 3.1, and Table 6 in the Appendix), 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).

The network devices are evaluated in a lab environment by connecting them to an internet-enabled router (like in a domestic network setup) with the OpenWrt firmware [51]. 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); see Fig. 2 (a). Otherwise, the test device is connected to the parental control device's wireless AP; see Fig. 2 (b). We capture network traffic on both the test device and the router using Wireshark and tcpdump, respectively.

For Android apps, we maintain two experimental environments to concurrently record and inspect network traffic originating from the child and parent apps. We examine the child apps using a Samsung Galaxy S6 phone running Android 7.0; for the parent apps, we use a Nexus 4 with Android 5.1.1. We run a full Linux distribution with mitmproxy [48] and tcpdump on each experimental environment by installing Linux Deploy [7], and configured Android's network settings to proxy all traffic going through the WiFi adapter to the mitmproxy server. This enables us to capture the network traffic directly within the mobile devices.

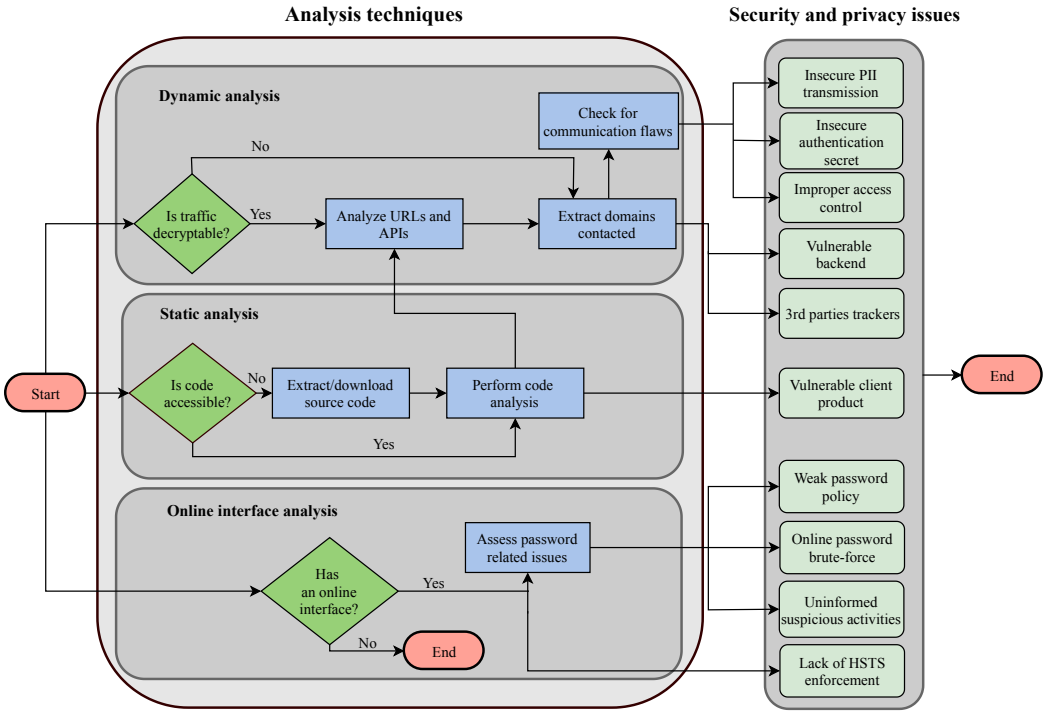
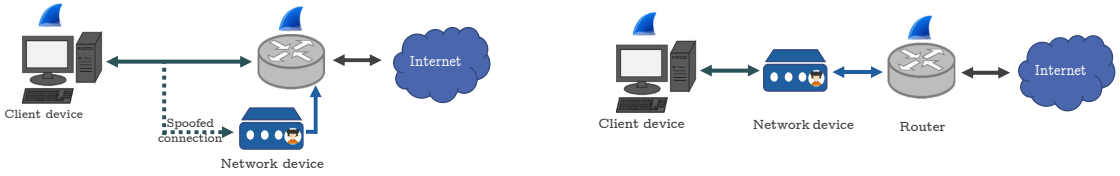


Figure 1: Overview of our evaluation framework.



(a) ARP poisoning case.

(b) Dedicated Access Point case.

Figure 2: Network devices test environment. Wireshark is installed on both the client device and home router.

We test each Windows application and Chrome extension on a fresh Windows 10 virtual machine with Chrome, tcpdump and mitmproxy installed. We intercept inbound and outbound traffic using mitmproxy on the host, and record packets using tcpdump.

4.1.2 Traffic Analysis. After intercepting traffic, we parse and commit the collected tcpdump traffic to an SQLite database and check for the following security and privacy related issues.

PII and authentication secrets leakage. We examine the collected traffic to check for PII and authentication secrets transmitted in plaintext, or leakage of PII to third-party domains. We create a list of possible PII (see Table 8 in the Appendix) that can be leaked via the Request URL, Referer, HTTP Cookie, requests' payload, and LocalStorage. 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.

Improper access control. We parse the traffic to find API endpoints with improper access control. First, we try to identify all the APIs that can be potentially exploited (without strong authentication), using Postman (postman.com) to replay the recorded HTTP request stripped of authentication headers (e.g., cookies and authorization header). Any request successfully replayed is labeled as potentially vulnerable (in a database). Afterwards, 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. For instance, we deem a device's access control insecure if its own MAC address is used for API endpoints authentication, as the MAC address can easily be found by an attacker on the local network.

Identifying trackers. We use the EasyList [21], EasyPrivacy [22], and Fanboy [23] to identify known trackers. We also add known

trackers from past work [58, 75] to our list. To identify third-party SDKs in the parental control tools traffic, we use the WHOIS [57] registration record to compare the SDK owner name to the parental control website owner. In cases where the SDK information is protected by the WHOIS privacy policy, we visit the SDK's domain to detect any redirect to a parent site; we then lookup the parent site's registration information. If this fails, we manually review the SDK's "Organization" in its TLS certificate, if available. Otherwise, we try to identify the SDK owner by searching in crunchbase.com.

4.1.3 Backend Assessment. Due to ethical/legal concerns, we refrain from using any invasive vulnerability scanning tools to assess backend servers. Instead, we 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. Additionally, we use the Qualys SSL Test (Qualys 2020: ssllabs.com) to evaluate the security of the SSL configuration of the parental control tools' backends.

4.1.4 Challenges. During the interception and traffic analysis phase, we encountered several challenges. We summarize them here, including the tools and techniques we use to address them.

Network traffic attribution. On Android apps, a key issue is to identify the process that generated the traffic in the absence of the packets' referral metadata. We test how the app behaves when the child uses her device normally (e.g., phone calls, messaging, browsing). These activities produce a large amount of traffic that we need to match to the corresponding processes. We use the mitmproxy addon [48] to call `netstat` to detect the process name for every packet. We directly use netstat from the underlying Linux kernel (in our Linux Deploy setup) to capture the process ID and process name as soon as a connection is created, while previous work [41, 59] read and parse the system `proc` directory from the Android Linux kernel by checking the directory periodically. This past approach misses connections that are opened and closed before the next time they check the `proc` directory, while our approach looks into the live connection as soon as a connection is created. We may only miss very short-lived connections that are not detected by netstat. To the best of our knowledge, we achieve more reliable traffic-process attribution compared to past work. We leave a full evaluation of the effectiveness of the technique to future work.

Traffic interception. Most network devices use TLS for communicating with their backends. This prevented us from inserting a root certificate on these devices, so some of the network traffic generated by them is completely opaque to us. In these cases, we rely on static analysis of the device's firmware. In cases where an Android app uses certificate pinning to refuse server certificates signed by any CA other than the pinned certificate in the app, we use SSLUnpinning [1] to attach several hooks in the SSL classes in order to bypass this feature and intercept the communication. In cases where the child app installs a VPN on the child device to filter and block websites, we intercept the traffic by deleting the VPN configuration from Android setting on the child device. If the app stops functioning without the VPN, we update the app configuration file whenever possible to disable the setup of the VPN on startup of the app on the child device. One Windows application, Qustodio uses its own encrypted certificate store, for which, we extract the associated TLS proxy private key by dumping the process memory.

It is possible that due to our employed measures for traffic analysis and attribution (e.g., rooted device, disabled VPN), some parental control solutions may have functioned differently, which is difficult to verify due to the use of heavily obfuscated code. Hence, our findings may be the lower-end of the actual privacy exposure.

4.2 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.

Network devices. We analyze the network device firmware whenever possible. We either attempt to extract the firmware directly from the device (via JTAG, UART, or ICSP interfaces), or download the device firmware from the vendor's website. We found 3/8 network devices with an accessible serial UART port (KoalaSafe, Blocksi, and Fingbox) that we used to extract the firmware from the devices. Another device (Circle) made its firmware available online. Among the remaining devices (without access to their firmware), we scan for the presence of open remote admin services (e.g., SSH), which are often closed or key-protected. To identify vulnerable services, we scan the network devices with several tools (OpenVas, Nmap, Nikto [16] and Routersploit [61]), and match the identified software versions against public vulnerability databases.

Chrome extensions. We manually analyze the source code of the Chrome extensions, which mainly consists of scripts, separated into content scripts and background scripts. As most Chrome extensions' codebase is relatively small, and do not involve serious obfuscation, we can investigate their operations and detect security and privacy issues (e.g., PII leakage, common JavaScript vulnerabilities).

Android apps. We perform an automated analysis on all 153 Android apps using Firebase Scanner [63] to detect security misconfigurations in Firebase.[4] We also use LibScout [10] 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 [56] to classify tracking SDKs. We use MOBSF [49] to extract the list of third-party tracking SDKs from all 153 apps based on Exodus-Privacy's tracker list.

4.3 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.

SSLStrip attack. To check for SSLStrip attacks, we first set up a WiFi AP with mitmproxy, SSLStrip2 [40] and Wireshark installed. Then, we connect the parental control tool to our WiFi access point. Wireshark is utilized to record network traffic while mimicking common use case scenarios with the goal of triggering all parental control monitoring and control UI and API requests looking for

[4] Google Firebase (https://firebase.google.com/) provides support for backend infrastructure management for Android apps.

signs of successfully SSL Stripping attack on the traffic. We confirm the effectiveness of the attack by comparing the result to the corresponding traffic in a regular testing environment (i.e., without SSLStrip).

Weak password policy. During the parental control tool's account creation, we evaluate its password policy. We adopt a fairly conservative stance and only labelled as weak the password policy accepting password with 4 characters or less.

Online password brute-force. We use Burp Suite [55] to perform password brute-force attacks on our own online accounts. To keep the load on the server minimal, we test for the presence of defensive mechanisms by 50 attempts on our account from a single computer.

Uninformed suspicious activities. To determine whether the solution presents measures to report suspicious activities, we test two scenarios in which the user should be notified: modification of the user's password, and connection to the account from a new/unknown device. We deem a parental control tool that does not alert (e.g., via email) in either case to be vulnerable.

5 RESULTS

Following the methodology in Sec. 4, we analyzed the parental control tools between Mar. 2019 to May 2020, which include: 8 network devices, 29 Android apps representing 13 Android solutions, 10 Chrome extensions and 8 Windows applications. We also performed an automated analysis of 153 parental control Android apps to detect vulnerable backend databases and check for tracking SDKs. In this section, we report our findings on the tested security and privacy issues (as outlined in Sec. 3.3); for an overview, see Table 1.

5.1 Vulnerable Client Product

Network devices. The importance of securing the update mechanism has been known for years, cf. [11]. Surprisingly, 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. We confirmed this vulnerability to be exploitable. We also found another vulnerability that enables executing a command as root on the Blocksi device via command injection (i.e., unsanitized user input is passed directly to a system shell for execution). We confirmed this vulnerability to be exploitable by sending a `router_setGeneralSettings` request to the Blocksi API endpoint, and injecting a command in the timezone field in the request parameters. The settings change triggers a WebSocket Secure (WSS) message to the Blocksi device. The device then reads the new configuration from the API endpoint and updates its local configuration.[5]

We also found that KoalaSafe runs Dropbear v2014.63 SSH server/client (released on Feb. 19, 2014), associated with four known remote code execution vulnerabilities. Under certain conditions, the KoalaSafe device opens a reverse SSH tunnel through its backend server, exposing the vulnerable SSH Dropbear server to an attacker outside the local network. By calling a KoalaSafe API endpoint,[6] an external attacker can detect when a reverse SSH tunnel is open using only the victim device's MAC address. If the tunnel is open, the API endpoint responds with the tunnel's port number, 0 otherwise. For large-scale exploitation, an attacker can query the aforementioned API endpoint to enumerate all KoalaSafe devices with the reverse tunnel open. This enumeration is feasible as KoalaSafe uses the GuangLia network interface card (NIC), and MAC addresses assigned to GuangLia NICs [70] are limited to only 2^{20} values.

Android apps. We found 3/13 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.

We also found that Kidoz, KidsPlace, and MMGuardian use custom browsers to restrict and filter web content. The three browsers fail to enforce HSTS, and lack persistent visual indication if the website is served on HTTP. KidsPlace safe browser keeps the address bar that shows visited URL to help with visual identification. However, MMGuardian shows the URL in the address bar until the page is fully loaded and then the URL is replaced with the webpage title. Following our disclosure, MMGuardian removed their custom browser.

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, despite the enhanced collision attack on SHA1 [42]. Dr. Web also accepted Diffie-Hellman 1024 (considered weak [2], and deprecated in Safari and Chrome since 2016 [15]). In addition, none of the proxies rejected revoked certificates. We also found that upon uninstallation of these applications, the root certificates associated with the proxies remained in the Windows trusted root certificate store, with four of them having a validity duration over one year.

Two Chrome extensions (Adult Blocker and MateCode Blocker) download and run a third-party tracking script at run time. The domains hosting the scripts are not apparently related to the extension providers (or libraries from well-known companies). Note that runtime loaded scripts bypass 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 [34].

5.2 Vulnerable Backend

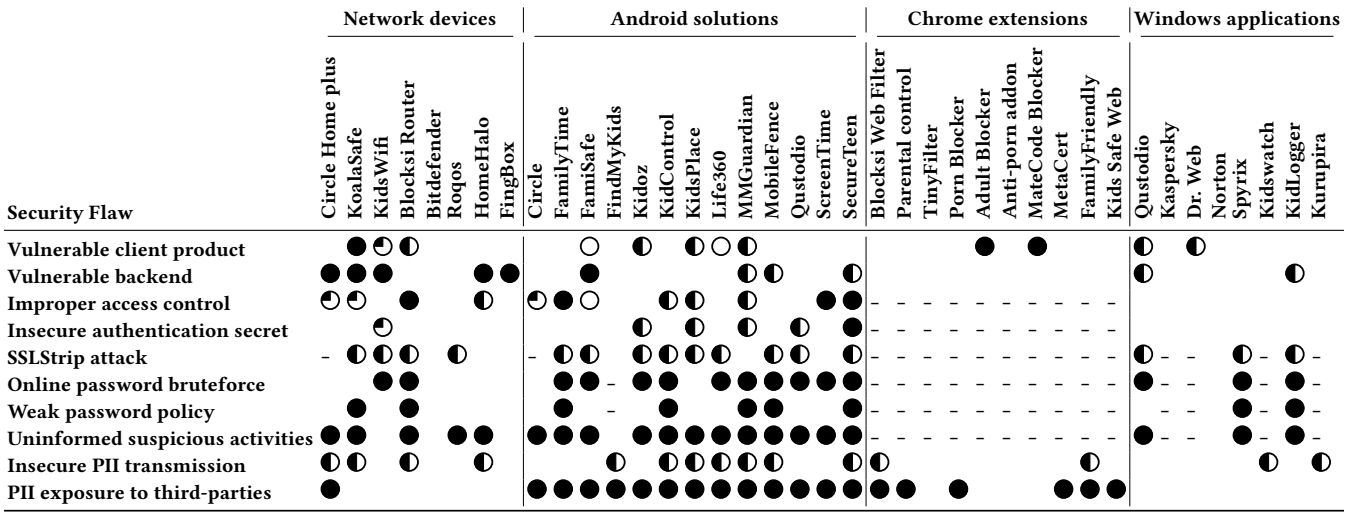
Network devices. Examples of vulnerable software components from our analysis of backend server API endpoints include: Apache 2.4.34 with 11 CVEs in KoalaSafe; PHP 7.0.27 with 26 CVEs in KidsWifi; Nginx versions with the same 3 CVEs in KidsWifi, Circle, HomeHalo and Fingbox. The Blocksi's API endpoint only indicates that it runs on OpenResty and Google Frontend (no version info).

Android apps. Since 115/153 Android apps use Google Firebase as a backend service, we analyzed their Firebase configuration for

[5] The timezone value is passed as `tz` to ["echo" + tz + "> /etc/TZ"]. Thus, if `tz` is "$(ls)", the `ls` command would be executed and its output written to /etc/TZ.

[6] https://api.koalasafe.com/api/router/[MACaddress]/et

Table 1: Overall results for security flaws in parental control tools labelled following the threat model in Sec. 3.2. ○ : 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.



security issues by performing an automated analysis using Firebase Scanner [63]. Critical misconfigurations can allow attackers to retrieve all the unprotected data stored on the cloud server. We followed a similar approach to Appthority's work [9] on scanning apps for Firebase misconfigurations. We found 8/153 Android apps with insecure Firebase configurations. We then evaluated the type of sensitive data exposed by each app to determine the impact of the data being leaked. For ethical reasons and to protect other customers privacy, we created a parental account on the eight apps. Then, we updated the Firebase scanner to automatically search for our test data in the its response and record the leaked information from our own account. We found three apps exposing personal information: 1) FamiSafe with 500K+ installs exposes the parent email; 2) Locate with 10K+ installs exposes the child name, phone number, and email; and 3) My Family Online with 10K+ installs exposes the child name, child and parent phone numbers, parent email, and apps installed on child phone. FamiSafe fixed the Firebase security issue following our disclosure. Additionally, we found that MMGuardian, MobileFence, and SecureTeen servers support RC4, and SecureTeen backend is vulnerable to the POODLE attack.

Windows applications. We found that some Windows applications' servers also do not use ideal TLS configurations. For instance, Qustodio's server has an intermediate certificate signed with SHA1 in its chain of trust. Qustodio and KidLogger servers support the RSA key exchange protocol which lacks forward secrecy.

5.3 Improper Access Control

Network devices. The KoalaSafe API login endpoint requires three parameters that are available to anyone on the local network: a device-generated authentication token, the device's date and time, and the device's MAC address for successful authentication. These parameters can be obtained by visiting endpoints hosted by the KoalaSafe device.[7] Thus, a local network attacker can easily collect the information needed for authentication and use the API endpoint to access sensitive information such as the profile name, email address, and browsing history.

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. This is because a remote attacker can retrieve a user's email using their device SN or vice-versa.[8] By sending both parameters to the API endpoint in a POST message, any remote attacker can authenticate to the server, and access sensitive information about the home network, e.g., the WiFi password, and MAC addresses of connected devices.

The HomeHalo device uses only the device's SN and an HTTP header called `secretToken` to authenticate to its API endpoint. In our case, the secretToken had a fixed value of 100500. An on-path attacker can intercept and modify these messages, and gain access to admin controls, e.g., reading or changing the wireless SSID, password, or even the device's root password. Other privacy sensitive information is also exposed, including: the devices connected to HomeHalo's network and the parental control profile setup.

The Circle Home Plus creates a profile for each child and stores it locally on the device, including the child age groups, usage history and statistics, child photo, and username (i.e., some parents may use child name). We identify two API endpoints used to transmit child information in plaintext over the local network. The first API endpoint[9] sends child account usage history and statistics, and `profileID`. It insecurely relies on the requester's MAC address to

[7] Authentication token and device time are available at https://device.koalasafe.com/auth.lua, and the MAC address at https://device.koalasafe.com/status.lua

[8] For SN to email, use https://service.block.si/config_router_v2/router_checkRouters/null/[SN], and for email to SN, https://service.block.si/config_router_v2/router_checkRouters/[email].

[9] http://10.123.234.1/api/USERINFO?host=ios&nocache=1572292313630HTTP/1.1

identify the child device and communicate sensitive information. This API endpoint is called whenever a child device attempts to access a restricted domain. The second API endpoint[10] fetches the profile photo corresponding to the received profile ID.

Android apps. We found 8/13 Android solutions lack authentication for accessing PII. Prominent examples include the following. In FamilyTime, a six-digit parameter `childID` is generated through a sequential counter incremented by one per user. An attacker can retrieve the child name, gender, date of birth, email address, and child phone number through an API request that requires only the childID value. Hence, an attacker can remotely exploit this vulnerability at a large scale, simply by trying all 6-digit values.[11] In FamiSafe, an attacker can retrieve all the child social media messages and YouTube activities labeled as suspicious through an API request that requires the following parameters: `deviceid`, `memberid`, `client_sign`, and `access token`. However, any app installed on the child device can access these parameters from the FamiSafe log file on the shared external storage.[12]

5.4 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. We deem this a minor risk as the vulnerability is only present for a limited duration (during device setup), and the attacker must be within close proximity.

Android apps. In SecureTeen, we found an API endpoint that can be used to authenticate the user to the parental control account. This API endpoint enables any adversary to remotely compromise any parental account by knowing only the parent's email. When the API request is invoked by the browser, the adversary is logged in to the parental dashboard and obtains full access to the parent account, including the ability to monitor and control the child device.[13]

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, with validity periods of one year and two hours respectively. Even with the 2-hour cookie in Qustodio, the attacker can easily access sensitive information about the child including the child's current location, and history of movements. The attacker can also access remote control functions on the child phone, such as block all incoming/outgoing calls. In the case of KidsPlace, the attacker can access a wide spectrum of remote control functions to the child phone such as: disable the Internet, silently install a malicious app on the child device, or upload harmful content to the child mobile. The attacker can also lock the child phone making her unable to contact the parent or perform an emergency call.

5.5 SSLStrip and Online Account Issues

We found that nine Android solutions, four network devices and three Windows applications transmitted the parent account credentials via HTTP under an SSLStrip attack. This allows an adversary to compromise the parent account for a long time, particularly if the app does not send any notification to the parent when the account is accessed from a new device. More seriously, in Kidoz, we could see the parent's credit card account number and email in HTTP when using their BlueSnap online payment solution [12], while connected to our WiFi access point. This was possible because the online payment server is not configured to use HSTS. In Qustodio, we could extract the child Facebook credentials provided by the parent during the configuration of the monitoring component. Following our disclosure, FamilyTime enabled HSTS on their server.

In terms of defense against online password guessing, we found that two network devices and 10 Android solutions leave their online login interfaces open to password brute-force attacks. Also, two network devices, five Android solutions, and three Windows applications enforced a *weak* password policy (i.e., shorter than four characters). We also observed that five network devices, 12 Android solutions and four Windows applications do not report suspicious activities on the parent's account such as password changes and accesses from unrecognized devices. These activities are possible indicators of account compromise and should be reported to the user.

5.6 Insecure PII transmission

Network devices. 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. This can allow on-path attackers to track the child's web activities [18]. The HomeHalo device suffers from a similar problem: whenever a domain is requested by a user device inside its network, HomeHalo sends an HTTP request, including the child device's MAC address, to its backend server to identify the requested domain's category.

Android apps. Several Android solutions send cleartext PII, see Table 9 in the Appendix. Examples include: 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). MMGuardian transmits the child visited URL (Base64 encoded) to a third-party domain classifier Komodia.com [39]) via HTTP. When we contacted MMGuardian, they informed us that they are working with Komodia on a resolution. Other products using Komodia are also apparently affected by this.

Windows application and Chrome extensions. During the installation phase of Kurupira, the user has to set up an SMTP server with the assistance of the application to receive activity reports. However, in case the user uses an SMTP server with an unencrypted protocol, Kurupira does not warn about transmitting child activity report in plaintext. Kidswatch sends child activity reports over HTTP. We also found that three extensions (Blocksi Web Filter, FamilyFriendly Parental Control, Porn Blocker) send the domain

[10] http://10.123.234.1/api/USERPHOTO?profileID=[profileID].

[11] By using e.g., a cURL commad (the last parameter is childID): $ curl -v https://mesh.familytime.io/v2/child/Android/profile/456***.

[12] By using e.g., a cURL command: $ curl -v https://u.famisafe.com/load-page/index?page=suspicious-text/detail&access_from=1&device_id=165***&member_id=1045***&client_sign={fffff***-be**-19ec-0000-000075b3****}&access_token=dtwMtFarI********&lang=en.

[13] An example call to the API is as follows: https://cp.secureteen.com/auth.php?&productName=secureteen&resellerId=careteen&page=menu&loginFromApp=Yes&j_username=parentemail**@gmail.com&gType=monitoring.

contacted by the user to the extension's server using HTTP to check whether or not the website should be blocked.

5.7 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. In Android, we found use of trackers in most apps via static analysis, including: the children apps (targeted for children's devices only, 44/51 apps with tracking SDKs), shared apps (the same app is used by both parents and children, 73/78 apps), and parent apps (targeted for parents' devices only, 22/24 apps); see Table 7 in the Appendix. Over 25% of children apps utilize advertising networks (e.g., Google Ad and Doubleclick SDKs; see Fig. 3 in the Appendix) which could potentially violate US COPPA. For network devices, our static analysis for five companion apps reveals the use of tracking SDKs (2–12 unique trackers) in all those apps except for KoalaSafe. For Chrome extensions, we found that half of the Chrome extensions send behavioral information (e.g. web browser usage) to Google Analytics.

We also identify tracking third-party SDKs from network traffic generated during our dynamic analysis from child device. Except SecureTeen, 12/13 Android solutions use tracking SDKs (1–16 unique trackers; see Fig. 4 in the Appendix). 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.[14] We also found that one of the network devices' companion app, Circle, includes a third-party analytical SDK from Kochava. Every time the app is launched, or it returns to the foreground, the following information is shared with Kochava: Device ID (enables tracking across apps), device data (enables device fingerprinting for persistent tracking). Kochava provides an opt-out option (`app_limit_tracking=true`) that can be used to comply with COPPA. However, the Circle transmits this flag as `false` from the child device.[15]

For Android solutions that have a safe custom browser, such as Kidoz, MMGuardian, and KidsPlace, we found that all these browsers allow visited websites to store persistent tracking HTTP cookies (or Local Storage) on the child device. These cookies are not erased when the browser app is closed.

Restricted SDKs from past work. We also study the SDKs identified in past studies [28, 62] 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). We evaluated the privacy policies for the seven prohibited SDKs detected, and concluded that four companies, Crashlytics, Amplitude, Braze (formerly Appboy), and Appnext, still prohibit the use of their SDKs in children's apps; two others (Tapjoy and Branch) now require developers to set the appropriate parameters; and the last one (Supersonic/ironSource) removed any restriction.

From static analysis, we found several prohibited SDKs being used: 25/44 children apps and 8/73 shared apps use Google CrashLytics; unGlueKids children app uses Branch SDK (without the `do-no-track` mode); and Limitly uses Appnext SDK. Aside from Google CrashLytics, we also observed Branch (7 apps), Amplitude (6 apps), Braze (4 apps), and Tapjoy (1 app) SDKs in the shared apps.

Through analysing traffic generated from child device, we confirm that five Android solutions use prohibited SDKs. Also for Life360, we note that Branch SDK "do-not-track" mode was disabled since the network traffic from child device contains Android ID, Android Advertising ID (AAID), and local private IP. Additionally, three Android solutions FindMyKids, KidsPlace, and Circle contact Crashlytics prohibited SDK server (reports.crashlytics.com), and Qustodio communicates with the Braze prohibited SDK.

PII exposure to third-parties. We found that all Android solutions share personal and unique device information with third-party domains (see Table 10 in the Appendix). Prominent examples include: ScreenTime shares the child Android ID with Facebook. Four extensions send the requested domains to their server to check whether the website should be blocked, which can also be locally performed similar to Google Safe Browsing. More concerning 2/10 extensions send the complete URL, possibly leaking personal information not required for blocking. Another extension, Parental Control, overrides Chrome setting and replaces the default search URL by its server domain, which automatically redirects to Google Safe Search, but exposes the search terms to the extension's server. We also found that another Chrome extension, Porn Blocker, redirects the user to https://www.purplestats.com/page/blocked/ when visiting a blocked website, and leaks the full URL of the previous webpage through the referer header.

COPPA Safe Harbor providers. We check the behavior of (3/13) (Kidoz, FamilyTime, FindMyKids) Android solutions certified by the US FTC's COPPA Safe Harbor program [74] (by kidSAFE [65]; we also checked other programs under Safe Harbor, and the parental control tools websites/descriptions). Our traffic analysis collected from the child device reveals that FindMyKids use three trackers and leak Android Advertising ID to at least two trackers graph.facebook.com and adjust.com. FindMyKids includes two flags when calling Facebook to enable application tracking and advertiser tracking (both were enabled) [26]. FindMyKids also shares child Android ID with Yandex Metrica (appmetrica.yandex.net). Yandex Metrica provides an option (`limit_ad_tracking`) that can be used to restrict tracking. However, the FindMyKids transmits this flag as `false` from the child device [80]. We also found that FamilyTime sends the child's name, email address and phone # (hashed in SHA256) to facebook.com. Kidoz uses eight trackers and leaks the Android Advertising ID to the third-party domain googleapis.com through the referer header.

6 POTENTIAL PRACTICAL ATTACKS

In this section, we summarize the impact of exploiting some of the discovered vulnerabilities in the analyzed parental control tools.

Device compromise. Device compromise presents serious security and privacy risks, especially if a vulnerability can be exploited

[14] The use of `tfcd=1` marks an ad request as child-directed; see https://support.google.com/admanager/answer/3671211?hl=en.

[15] Note that Disney, a former partner of Circle, is the target of a class action lawsuit for using a similar SDK in children's apps; see https://unicourt.com/case/pc-db1-rushing-et-al-v-the-walt-disney-company-et-al-494632.

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 (see Sec. 5.1 and Sec. 5.3). 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 [71]), or use it in a botnet (cf. Mirai [8]).

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 [36]); some products accept passwords as short as one character. Third, products with broken authentication allow access to parental accounts without credentials. For example, SecureTeen provides an API endpoint (see Sec. 5.4) 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. 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.

PII on the network. COPPA mandates reasonable security procedures for protecting children's information [32]. However, we found several parental control tools transmit PII insecurely. For example, FindMyKids leaks surrounding voice, and the child's picture. 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 recognize the child from her photo. KidControl allows the child to send SOS messages when she is in a dangerous situation. However, an attacker can identify and 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.

7 CONCLUSION

Parental control solutions are used by parents to help them protect their children from online risks. Nevertheless, some of these solutions have made news in the recent years for the wrong reasons. Our cross-platform comprehensive analysis of popular solutions shows systematic problems in the design and deployment of *all* the analyzed solutions (except Bitdefender, TinyFilter, Anti-porn addon, Kaspersky, and Norton) from a security and privacy point of view. Indeed several of these solutions can undermine children's online and real-world safety. As these solutions are viewed as an essential instrument to provide children a safer online experience by many parents, these solutions should be subjected to more rigorous and systematic evaluation, and more stringent regulations.

ACKNOWLEDGMENTS

This work was partly supported by a grant from the Office of the Privacy Commissioner of Canada (OPC) Contributions Program. We thank the anonymous ACSAC 2020 reviewers for their insightful suggestions and comments.

REFERENCES

[1] ACPM. 2016. SSLUnpinning - Xposed Module. https://github.com/ac-pm/SSLUnpinning_Xposed/.
[2] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. 2015. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *ACM CSS*. 5–17.
[3] Kendra Allison. 2018. *Online Risks, Sexual Behaviors, And Mobile Technology Use In Early Adolescent Children: Parental Awareness, Protective Practices, And Mediation.* Ph.D. Dissertation. University of South Carolina.
[4] Collin Anderson, Masashi Crete-Nishihata, Chris Dehghanpoor, Ron Deibert, Sarah McKune, Davi Ottenheimer, and John Scott-Railton. 2015. Are the Kids Alright? Digital Risks to Minors from South Korea's Smart Sheriff Application. Article.
[5] Android. Last accessed Oct. 2020. Android Device Administration. https://developer.android.com/guide/topics/admin/device-admin/.
[6] Android. Last accessed Oct. 2020. UI/Application Exerciser Monkey. https://developer.android.com/studio/test/monkey.html.
[7] Anton Skshidlevsky. Last accessed Oct. 2020. Linux Deploy. https://github.com/meefik/linuxdeploy/.
[8] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *USENIX Security*. 1093–1110.
[9] Appthority. 2018. Appthority: ENTERPRISE MOBILE THREAT REPORT - Firebase Vulnerability: Exposing Sensitive Data via Thousands of Mobile Apps.
[10] Michael Backes, Sven Bugiel, and Erik Derr. 2016. Reliable third-party library detection in android and its security applications. In *ACM SIGSAC CCS*. 356–367.
[11] Anthony Bellissimo, John Burgess, and Kevin Fu. 2006. Secure Software Updates: Disappointments and New Challenges. In *USENIX HotSec*.
[12] Bluesnap.com. Last accessed Oct. 2020. BlueSnap: Online Payment Solutions. https://home.bluesnap.com/.
[13] C. Marshall and C. Ellis. 2019. The best free parental control software 2019. https://www.techradar.com/news/the-best-free-parental-control-software/.
[14] Quan Chen and Alexandros Kapravelos. 2018. Mystique: Uncovering information leakage from browser extensions.. In *ACM SIGSAC CCS*. 1687–1700.
[15] Chrome. 2017. Remove DHE-based ciphers. https://www.chromestatus.com/feature/5128908798164992.
[16] CIRT.net. Last accessed Oct. 2020. Nikto Web Server Scanner. https://cirt.net/Nikto2/.
[17] Common Sense Media and SurveyMonkey. 2017. Think You Know What Your Kids Are Doing Online? Think Again. Survey report, https://www.commonsensemedia.org/blog/think-you-know-what-your-kids-are-doing-online-think-again.
[18] Mathieu Cunche. 2014. I know your MAC address: targeted tracking of individual using Wi-Fi. *Journal of Computer Virology and Hacking Techniques* (2014).
[19] Xavier de Carné de Carnavalet and Mohammad Mannan. 2016. Killed by proxy: Analyzing client-end TLS interception software. In *NDSS*.
[20] DQinstitute.org. 2020. Nearly two-thirds of children surveyed around the world are exposed to cyber risks, first-ever global Child Online Safety Index reveals. Online article. https://www.dqinstitute.org/news-post/nearly-two-thirds-of-children-surveyed-around-the-world-are-exposed-to-cyber-risks-first-ever-global-child-online-safety-index-reveals/.
[21] EasyList. Last accessed Apr. 24, 2020. https://easylist.to/easylist/easylist.txt.
[22] EasyList. Last accessed Apr. 24, 2020. https://easylist.to/easylist/easyprivacy.txt.
[23] EasyList. Last accessed Apr. 24, 2020. https://easylist.to/easylist/fanboy-social.txt.
[24] Ronald Eikenberg. 2019. Kaspersky script injection. https://www.heise.de/ct/artikel/Kasper-Spy-Kaspersky-Anti-Virus-puts-users-at-risk-4496138.html.
[25] EPIC.org. 2010. FTC Settles with Company that Failed to Tell Parents that Children's Information Would be Disclosed to Marketers.

https://www.ftc.gov/news-events/press-releases/2010/11/ftc-settles-company-failed-tell-parents-childrens-information.
[26] Facebook.com. Last accessed Oct. 2020. App Events API. https://developers.facebook.com/docs/marketing-api/app-event-api/.
[27] Facebook.com. Last accessed Oct. 2020. Manually Build a Login Flow. https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow/.
[28] Álvaro Feal, Paolo Calciati, Narseo Vallina-Rodriguez, Carmela Troncoso, and Alessandra Gorla. 2020. Angel or Devil? A Privacy Study of Mobile Parental Control Apps. In *PETS*.
[29] OWASP Foundation. Last accessed Oct. 2020. HTML5 Security Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#local-storage.
[30] OWASP Foundation. Last accessed Oct. 2020. REST Security Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html.
[31] FTC.gov. 2014. Parental Controls. Online article. https://www.consumer.ftc.gov/articles/0029-parental-controls.
[32] FTC.gov. Last accessed Oct. 2020. 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.
[33] Google. 2012. Implementing OAuth 2.0 Authorization. https://developers.google.com/youtube/v3/guides/authentication.
[34] Google. Last accessed June 2020. Multiple Extensions are Compromised in a Browser Hijacking Scam. https://support.google.com/chrome/thread/46798301.
[35] Google. Last accessed Oct. 2020. Google Firebase. https://firebase.google.com/.
[36] Paul A Grassi, Ray A Perlner, Elaine M Newton, Andrew R Regenscheid, William E Burr, Justin P Richer, Naomi B Lefkovitz, Jamie M Danker, and Mary F Theofanos. 2017. *Digital identity guidelines: Authentication and lifecycle management [including updates as of 12-01-2017].* Technical Report.
[37] IETF.org. Last accessed Oct. 2020. HTTP Strict Transport Security (HSTS). https://tools.ietf.org/html/rfc6797.
[38] Jon Martindale. 2019. Keep your kids safe online with these great parental control tools. Article. https://www.digitaltrends.com/computing/best-free-parental-control-software/.
[39] Komodia.com. Last accessed Oct. 2020. Komodia URLs classification SDK. https://url-classification.io/industry/parental-control/.
[40] L. Nve. Last accessed Oct. 2020. SSLStrip2. https://github.com/LeonardoNve/sslstrip2/.
[41] Anh Le, Janus Varmarken, Simon Langhoff, Anastasia Shuba, Minas Gjoka, and Athina Markopoulou. 2015. AntMonitor: A system for monitoring from mobile devices. In *ACM SIGCOMM C2B(I)D.*
[42] Gaëtan Leurent and Thomas Peyrin. 2019. From collisions to chosen-prefix collisions application to full SHA-1. In *EUROCRYPT*. Springer.
[43] Xurong Li, Chunming Wu, Shouling Ji, Qinchen Gu, and Raheem Beyah. 2017. HSTS Measurement and an Enhanced Stripping Attack Against HTTPS. In *SecureComm*. Springer.
[44] Meng Luo, Pierre Laperdrix, Nima Honarmand, and Nick Nikiforakis. 2019. Time Does Not Heal All Wounds: A Longitudinal Analysis of Security-Mechanism Support in Mobile Browsers. In *NDSS*.
[45] Jack Madden and Brian Madden. 2013. *Enterprise Mobility Management: Everything you need to know about MDM, MAM, and BYOD.* Jack Madden.
[46] Mark Jones, Komando.com. 2018. Parental control app database exposed, leaving kids' information compromised. News article. https://www.komando.com/happening-now/461381/parental-control-app-database-exposed-leaving-kids-information-compromised.
[47] Abigail Marsh. 2018. *An Examination of Parenting Strategies for Children's Online Safety.* Ph.D. Dissertation. Carnegie Mellon University.
[48] MITMProxy.org. Last accessed Oct. 2020. HTTPS proxy. https://MITMProxy.org/.
[49] MOBSF. Last accessed Oct. 2020. Mobile-Security-Framework-MobSF. https://github.com/MobSF/Mobile-Security-Framework-MobSF.
[50] Brendan Moran, Hannes Tschofenig, David Brown, and Milosch Meriac. 2019. *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.
[51] OpenWrt Project. Last accessed Oct. 2020. OpenWrt. https://openwrt.org/.
[52] PCMag.com. 2019. The Best Parental Control Software for 2019. https://www.pcmag.com/article2/0,2817,2346997,00.asp/.
[53] Pew Research Center. 2016. Parents, Teens and Digital Monitoring. Survey report, http://www.pewinternet.org/2016/01/07/parents-teens-and-digital-monitoring/.
[54] Pierluigi Paganini. 2018. Parental control spyware app Family Orbit hacked, pictures of hundreds of monitored children were exposed. News article. https://securityaffairs.co/wordpress/75888/data-breach/family-orbit-hacked.html.
[55] Portswigger.net. Last accessed Mar. 2020. The Burp Suite family. https://portswigger.net/burp.
[56] Exodus Privacy. Last accessed Mar. 2020. The privacy audit platform for Android applications. https://reports.exodus-privacy.eu.org/en/trackers/.
[57] Pypi.org. Last accessed Oct. 2020. Python WHOIS Library. https://pypi.org/project/whois/.
[58] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, and Christian Kreibich Phillipa Gill. 2018. Apps, trackers, privacy, and regulators. In *NDSS*.
[59] Abbas Razaghpanah, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Christian Kreibich, Phillipa Gill, Mark Allman, and Vern Paxson. 2015. Haystack: In situ mobile traffic analysis in user space. *arXiv preprint arXiv:1510.01419* (2015).
[60] Bradley Reaves, Jasmine Bowers, Nolen Scaife, Adam Bates, Arnav Bhartiya, Patrick Traynor, and Kevin RB Butler. 2017. Mo (bile) money, mo (bile) problems: Analysis of branchless banking applications. *ACM TOPS* (2017).
[61] Reverse Shell Security. Last accessed Oct. 2020. Routersploit Embedded Devices Exploitation framework. https://github.com/threat9/routersploit/.
[62] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. 2018. "Won't somebody think of the children?" examining COPPA compliance at scale. *PETS* (2018).
[63] Shiv Sahni. Last accessed Mar. 2020. FireBase Scanner. https://github.com/shivsahni/FireBaseScanner.
[64] Samet Privacy, LLC. Last accessed Oct. 2020. Official membership page. https://www.kidsafeseal.com/certifiedproducts/kidoz_sdk_app.html.
[65] Samet Privacy, LLC. Last accessed Oct. 2020. Official membership page. https://www.kidsafeseal.com/certifiedproducts/familytime_app.html.
[66] Sellcell.com. 2019. Kids Cell Phone Use Survey 2019 – Truth About Kids & Phones. News article. https://www.sellcell.com/blog/kids-cell-phone-use-survey-2019/.
[67] Zhiyong Shan, Raina Samuel, and Iulian Neamtiu. 2019. Device Administrator Use and Abuse in Android: Detection and Characterization. In *MobiCom*.
[68] Sharon Shasha, Moustafa Mahmoud, Mohammad Mannan, and Amr Youssef. 2019. Playing with danger: A taxonomy and evaluation of threats to smart toys. *IEEE Internet of Things Journal* (2019), 2986–3002.
[69] Oleksii Starov and Nick Nikiforakis. 2017. Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions.. In *WWW*.
[70] Stiller, Nate. Last accessed Oct. 2020. MAC Address Lookup. https://www.macvendorlookup.com/mac-address-lookup/.
[71] Nikhil Tripathi and Neminath Hubballi. 2015. Exploiting dhcp server-side ip address conflict detection: A dhcp starvation attack. In *IEEE ANTS*. IEEE, 1–3.
[72] UK Council for Child Internet Safety (UKCCIS). 2016. Child Safety Online: A Practical Guide for Providers of Social Media and Interactive Services. Online article. https://www.gov.uk/government/publications/child-safety-online-a-practical-guide-for-providers-of-social-media-and-interactive-services.
[73] Unicef. 2020. Children at increased risk of harm online during global COVID-19 pandemic. Press release. https://www.unicef.org/press-releases/children-increased-risk-harm-online-during-global-covid-19-pandemic.
[74] U.S. Federal Trade Commission. Last accessed Oct. 2020. COPPA Safe Harbor Program. https://www.ftc.gov/safe-harbor-program/.
[75] Narseo Vallina-Rodriguez, Srikanth Sundaresan, Abbas Razaghpanah, Rishab Nithyanand, Mark Allman, Christian Kreibich, and Phillipa Gill. 2016. Tracking the trackers: Towards understanding the mobile advertising and tracking ecosystem. *arXiv preprint arXiv:1609.07190* (2016).
[76] Verizon. 2019. Verizon 2019 Data Breach Investigation Report. https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf.
[77] William Largent. 2017. Vulnerability Spotlight: The Circle of a Bug's Life. News article. https://blog.talosintelligence.com/2017/10/vulnerability-spotlight-circle.html.
[78] Wired.co.uk. 2019. A series of dumb security flaws left millions of EA Origin users exposed. News article. https://www.wired.co.uk/article/ea-origin-account-login-security-flaw.
[79] Pamela Wisniewski, Arup Kumar Ghosh, Heng Xu, Mary Beth Rosson, and John M Carroll. 2017. Parental Control vs. Teen Self-Regulation: Is there a middle ground for mobile online safety?. In *ACM CSCW*. 51–69.
[80] yandex.ru. Last accessed June 2020. AppMetrica tracking URL parameters. https://appmetrica.yandex.ru/docs/mobile-tracking/concepts/postback-specification.html.
[81] ZDNet.com. 2019. The latest dark web cyber-criminal trend: Selling children's personal data. News article. https://www.zdnet.com/article/the-latest-dark-web-cyber-criminal-trend-selling-childrens-personal-data/.

8 APPENDIX

In this appendix, we first provide some recommendations for parental control solution providers. Then, we present the corpus of parental control tools that we evaluated. Then, we provide a summary of the techniques adopted by the analyzed Android solutions to monitor child activities. Finally, we report our observations of tracking and PII sharing done by third-party SDKs and libraries embedded in these parental control tools.

8.1 Recommendations

In what follows, we list our recommendations for parental control solution providers.

Addressing vulnerabilities. Because of the sensitivity of the information manipulated by the parental control tools, companies should conduct regular security audits; the issues we listed in Sec. 3.3 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 participates 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 [29, 30]. 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 [76]. In the case of network devices, manufacturers should employ a secure firmware update architecture (see e.g., IETF [50]). 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 [37].

Limiting third-parties and SDKs. Parental control tools should limit the usage of trackers and tracking SDKs in apps intended for children. For the SDKs that allow special parameter for children's apps, those parameters must be used appropriately.

8.2 Parental Control Solutions Corpus

Tables 2, 3, 4 and 5 provide some information about the corpus of the parental control solutions analyzed throughout our study.

Table 2: List of parental control devices and their firmware versions.

Device	Version
Circle Home Plus	3.10.0.2
KoalaSafe	1.26825
KidsWifi	1.165
Blocksi Router	2.4
Bitdefender	2.1.66.4
Roqos	1.30.24
HomeHalo	1.0.0.8
Fingbox	0.5-2ubuntu4

Table 3: List of parental control Android solutions. ⋆ denotes versions downloaded from vendor websites with extra features; P, T refer to premium and trial versions, respectively.

Solution	Installs	App package name	Version
Circle	10K+	com.meetcircle.circle	P2.8.0.2
FamilyTime	500K+	io.familytime.dashboard	P2.1.0.210
		io.familytime.parentalcontrol	P3.0.5.3196.ps
		io.familytime.parentalcontrol(⋆)	P4.0.6.4209.web
FamiSafe	500K+	com.wondershare.FamiSafe	P3.0.9.107
FindMyKids	10M+	org.findmykids.app	T1.9.9
		org.findmykids.child	T1.9.9
Kidoz	1M+	com.kidoz	P4.0.5.8
		com.kidoz.demo.go(⋆)	P4.0.6.3
KidControl	1M+	ru.kidcontrol.gpstracker	T4.0.9
		app.gpsme	Tk5.2.10
KidsPlace	5M+	com.kiddoware.kidsplace	P3.5.6
		com.kiddoware.kidsafebrowser	P1.7.8
		com.kiddoware.kidsvideoplayer	P1.7.8
		com.kiddoware.kidsplace.remotecontrol	P1.4.5
		com.kiddoware.kidspictureviewer	P1.0.9
Life360	50M+	com.life360.Android.safetymapd	T18.7.1
MMGuardian	1M+	com.mmguardian.parentapp	P3.6.4
		com.mmguardian.childapp	P3.7.7
		com.mmguardian.childapp(⋆)	P10003.9.5
MobileFence	1M+	com.mobilefence.family	T2.9.3.1
		com.mobilefence.family.plugin(⋆)	T1.4
Qustodio	1M+	com.qustodio.qustodioapp	T180.14.2.2-family
		com.qustodio.qustodioapp(⋆)	T680.14.2.2-family
ScreenTime	1M+	com.screentime.rc	T3.11.23
		com.screentime	T5.3.23
SecureTeen	1M+	com.infoweise.parentalcontrol.secureteen	T8.0.0
		com.infoweise.parentalcontrol.secureteen.child	T1.6000.5
		com.infoweise.parentalcontrol.secureteen.child(⋆)	T1.7001.0

Table 4: List of parental control Windows applications and their corresponding websites' popularity (traffic/ranking); we analyzed the lasted versions available.

Application	# of visits/day	Main countries	World Alexa rank
Qustodio	27K	US	85,673
Kaspersky	1,400K	IN, US, RU	2,114
Dr. Web	84K	US	40,515
Norton	6,400K	US	431
Spyrix	21K	UK	230,966
Kidswatch	NA	NA	2,175,932
KidLogger	4.2K	PE	156,645
Kurupira	17K	BR	84,918

Table 5: List of parental control Chrome extensions.

Extension	Installs	ID	Version
Blocksi Web Filter	40K+	pgmjaihnmedpcdkjcgigocogcbffgkbn	1.0.144
Parental control	3K+	bdjgolepmhcchlgncgkmobepknekjbkd	1.0.22
TinyFilter	20K+	epniipcfpbjliciholgdeipceecgcfmj	2016.11.1.1
Porn Blocker	10K+	kmillccnmojidmkhhjngjlalnbhpobcl	1.5. 2
Adult Blocker	80K+	onjjgbgnpbedmhbdoikhknhflbfkecjm	6.2.8
Anti-porn addon	20K+	peocghcbolghcodidjgkndgahnlaecfl	2.20.0
MateCode Blocker	80K+	gppopmmjibhcboobpmfombbkoehgicoh	1.0.5
MetaCert	20K+	dpfbddcgbimoafpgmbbjiliegkfcjkmn	0. 10.18
FamilyFriendly	7K+	epdelmeadnnoadlcalkmacoopocdafnp	0.9.0
Kids Safe Web	3K+	lakceedfffnfheaipjadbcndkldlplnd	1.0.7

8.3 Techniques Adopted by Android solutions

Table 6 provides a summary of some techniques adopted by the analyzed Android solutions. Four Android solutions (MMGuardian, MobileFence, Qustodio, and SecureTeen), distributed via their company websites, support additional features compared to their Google Play store version.

Table 6: Techniques used to monitor child activities including web filtering, phone calls, SMS, and social media. ●: refers to service supported by Google Play version; ◐: refers to a feature supported by a version distributed via the company website.

Technique	Circle	FamilyTime	FamiSafe[16]	FindMyKids	Kidoz	KidControl	KidsPlace	Life360	MMGuardian	MobileFence[17]	Qustodio	ScreenTime	SecureTeen[18]
Superuser request										●			
Activate device admin		●	●	●			●		●	●	●	●	●
Monitor user actions		●	●	●			●		●	●	●	●	●
Retrieve window content		●	●	●			●		◐	●	●	●	●
Observe typed text									◐				●
Turn on explore by touch									◐				
Enhanced web accessibility									◐				
App usage statistics		●	●	●	●		●		●	●	●	●	●
Setup VPN connection		●	●										●
Install custom browser					●		●		●				
Run MDM Agent										●			
Read history bookmarks									●	●	●	●	●
Komodia SDK									●				●
Make/manage phone calls			●			●			●	●	◐		●
SMS permission			◐						●	◐	◐		◐
Notification access				●					●	●			◐
Access fine location		●	●	●		●	●	●	●	●	●	●	●
Custom SMS app									●				
Facebook Login											●		
YouTube OAuth				●			●						
Custom video player					●		●						
Take screenshot													◐

8.4 Third-Parties Analysis Results

Table 7 shows the use of third-party tracking SDKs in the analyzed 153 Android apps. We used MOBSF [49] to extract the list of third-party tracking SDKs from all apps based on Exodus-Privacy's tracker list. On average, we found 4.5 SDKs per app (max 10 SDKs) in children apps. The average number of SDKs increases to about 5.3 SDKs per app in shared apps and parent apps. We also found Google Firebase Analytics, Google CrashLytics are present in over 50% of all types of apps; see Fig. 3. We also identified tracking third-party SDKs from network traffic generated during our dynamic analysis; see Fig. 4.

[16]FamiSafe Android app gets full access to the child's YouTube account including rights to view, edit, delete the child's YouTube videos and playlists, and rate videos, post, edit/delete comments and captions.
[17]MobileFence initially setup by default to monitor both the child and parent devices.
[18]SecureTeen Android app uses a keylogger to record all social media activities on the child device.

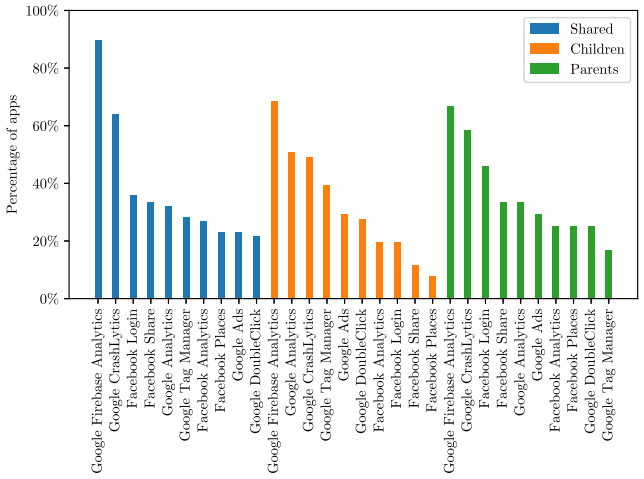


Figure 3: Tracking SDKs present in Android apps found through static analysis, see Sec. 5.7.

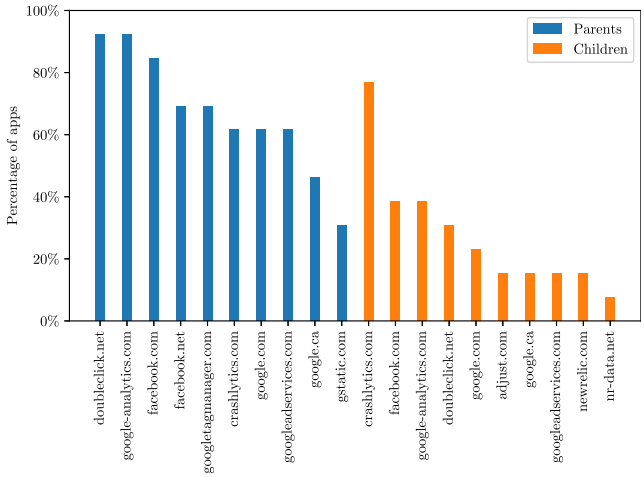


Figure 4: Tracking SDKs present in Android solutions found through dynamic analysis, see Sec. 5.7.

Table 7: Use of tracking SDKs in children apps, shared apps (i.e., the same is used by both parents and children), and parent apps found through static analysis.

	Children apps	Shared apps	Parent apps
# Android apps	51	78	24
# Unique tracking SDKs	35	41	31
# apps with tracking SDKs	44	73	22
Average # SDKs per app	4.5	5.3	5.4
Max # SDKs per app	10	22	12

8.5 Data Sharing and Privacy Leaks

Table 8 lists the personal information used to detect PII data in network traffic. Tables 9 and 10 show the PII transmitted by Android solutions in plaintext through HTTP, and PII shared with third-parties, respectively.

Table 8: The list of personal information used to detect PII data in network traffic.

PII	Description
AAID	Android Advertising ID
Android ID	Android ID generated on device setup
GSF ID	Google Services Framework ID
Phone Serial	Mobile serial number
IMEI	Phone equipment ID
SIM ID	SIM card ID
AP BSSID	MAC addresses of used hotspots
AP SSID	SSIDs of used hotspots
Nearby AP BSSID	MAC addresses of surrounding hotspots
Nearby AP SSID	SSIDs of surrounding hotspots
MAC Address	MAC address of the WiFi interface
IP address	IP address of the WiFi interface
BD ADDR	MAC address of the Bluetooth interface
Google Email	Google play account email address
User credentials	Account ID and password
Name	User's first and last names
Email	User's email address
Phone #	User's phone number
Geolocation	Latitude & Longitude
Contacts	Contact list entries
Browsing history	Visited URLs in browser
Used App	Apps used on the device
Installed Apps	Apps installed on the device
Social messages	SMS/social media messages
Search history	Search strings used on Google or Youtube
Mobile carrier	User's mobile carrier
Address	User's address (street name, city, country, and postal code)

Table 9: Android solutions sending sensitive data in plaintext.

Solution	Data	Destination
Kidoz	Account username/password	kidoz.net
Kidoz	Child name	kidoz.net
KidsPlace	Child Android ID	kiddoware.com
KidsPlace	Child phone serial	kiddoware.com
KidsPlace	Parent email	kiddoware.com
KidControl	Child Geolocation	kid-control.com
KidControl	Parent email	kid-control.com
KidControl	Parent name	kid-control.com
Life360	Child name	pubnub.com
Life360	Parent name	pubnub.com
MMGuardian	Account username/password *	mmguardian.com
MMGuardian	Parent email	mmguardian.com
MMGuardian	Parent IMEI	mmguardian.com
MMGuardian	Parent/child phone #	mmguardian.com
MMGuardian	Parent phone serial	mmguardian.com
MMGuardian	Browsing history	komodia.com
MMGuardian	Parent AAID	mmguardian.com
MMGuardian	Child Geolocation	mmguardian.com
MMGuardian	Child installed apps	mmguardian.com
SecureTeen	Parent email	secureteen.com
SecureTeen	Parent name	secureteen.com

*: The parent's password is hashed using SHA1 without salting.

Table 10: Sharing PII with third-parties.

Solution	Shared PII	3rd-parties (number, domains [max. 2]) *
Circle	Child Android ID	1 (kochava.com)
Circle	Parent Android ID	2 (kochava.com, mixpanel.com)
Circle	Child/parent AAID	1 (kochava.com)
Circle	Child/parent AP BSSID	1 (kochava.com)
Circle	Child/parent AP SSID	1 (kochava.com)
Circle	Child name	1 (intercom.com)
Circle	Parent email	5 (intercom.com, apptentive.com)
Circle	Parent name	3 (facebook.com, mixpanel.com)
Circle	Child mobile carrier	1 (kochava.com)
Circle	Parent mobile carrier	2 (kochava.com, apptentive.com)
FamilyTime	Child name	1 (facebook.com)
FamilyTime	Child email	1 (facebook.com)
FamilyTime	Child phone #	1 (facebook.com)
FamilyTime	Parent email	11 (doubleclick.net, facebook.com)
FamilyTime	Parent name	11 (fastspring.com, google-analytics.com)
FamilyTime	Parent address	1 (fastspring.com)
FamilyTime	Parent AAID	1 (facebook.com)
FamilyTime	Parent mobile carrier	1 (facebook.com)
FamilyTime	Parent phone #	1 (fastspring.com)
FamiSafe	Child AAID	1 (graph.facebook.com)
FamiSafe	Child name	1 (facebook.com)
FamiSafe	Child Geolocation	1 (maps.googleapis.com)
FamiSafe	Child browsing history	2 (facebook.com, google-analytics.com)
FamiSafe	Child device carrier	1 (graph.facebook.com)
Kidoz	Child AAID	1 (googleapis.com)
FindMyKids	Child/parent AAID	3 (yandex.net, facebook.com)
FindMyKids	Child/parent Android ID	1 (yandex.net)
FindMyKids	Child Geolocation	2 (openstreetmap.org, yandex.net)
FindMyKids	Child Nearby AP BSSID	1 (yandex.net)
FindMyKids	Child Nearby AP SSID	1 (yandex.net)
FindMyKids	Child/parent mobile carrier	1 (facebook.com)
KidControl	Child Geolocation	1 (openstreetmap.org)
KidControl	Parent email	1 (firestore.googleapis.com)
KidsPlace	Child AAID	2 (google-analytics.com, onesignal.com)
KidsPlace	Child mobile carrier	1 (onesignal.com)
KidsPlace	Child Geolocation	1 (maps.googleapis.com)
KidsPlace	Parent email	1 (sendgrid.com)
Life360	Child Android ID	1 (branch.io)
Life360	Parent Android ID	2 (branch.io, amazonaws.com)
Life360	Child AAID	3 (appsflyer.com, branch.io)
Life360	Parent AAID	4 (appsflyer.com, facebook.com)
Life360	Child/parent name	2 (braze.com, pubnub.com)
Life360	Child email	2 (helpshift.com, braze.com)
Life360	Parent email	3 (helpshift.com, braze.com)
Life360	Child/parent local IP	1 (branch.io)
Life360	Child/parent Geolocation	3 (locationiq.com, braze.com)
Life360	Parent phone #	1 (amazonaws.com)
Life360	Parent AP BSSID	1 (amazonaws.com)
Life360	Parent AP SSID	1 (amazonaws.com)
Life360	Child/parent mobile carrier	3 (appsflyer.com, braze.com)
MMGuardian	Child/parent AAID	2 (facebook.com, googleadservices.com)
MMGuardian	Child browsing history	1 (komodia.com)
MMGuardian	Child/parent mobile carrier	1 (facebook.com)
MobileFence	Parent email & name	1 (livechatinc.com)
MobileFence	Parent AAID	1 (googleadservices.com)
MobileFence	Child Geolocation	2 (googleapis.com, amazonaws.com)
MobileFence	Child browsing history	1 (google.com)
Qustodio	Child/parent Android ID	2 (amazonaws.com, rollout.io)
Qustodio	Child/parent AAID	1 (adjust.com)
Qustodio	Parent email	3 (adroll.com, braze.eu)
Qustodio	Parent name	1 (referralcandy.com)
Qustodio	Child used app	1 (google-analytics.com)
Qustodio	Child/parent mobile carrier	1 (braze.eu)
ScreenTime	Child/parent AAID	1 (graph.facebook.com)
ScreenTime	Child Android ID	4 (facebook.com, googleapis.com)
ScreenTime	Parent Android ID	1 (appspot.com)
ScreenTime	Child name	3 (appspot.com, facebook.com)
ScreenTime	Parent email & name	2 (appspot.com, facebook.com)
ScreenTime	Child Geolocation	4 (google.com, googleapis.com)
ScreenTime	Child installed apps	1 (appspot.com)
ScreenTime	Parent Mobile carrier	1 (facebook.com)
ScreenTime	Child/parent mobile carrier	1 (graph.facebook.com)
SecureTeen	Parent email	27 (adroll.com, ads.yahoo.com)
SecureTeen	Child browsing history	1 (komodia.com)
SecureTeen	Child Geolocation	1 (google.com)

*: Number of domains limited to 2 to fit display; AAID refers to Android Advertising ID; We use the word "domain" to refer to second-level domains.