Final Report for OPC Contributions Program 2019-2020

“Privacy Report Card for Parental Control Solutions”

Mohammad Mannan, and Amr Youssef
Student team members:
Suzan Ali, Mounir Elgharabawy and Quentin Duchaussoy

Concordia Institute for Information Systems Engineering (CIISE)
Concordia University, Montreal

Abstract

The digital age has introduced new challenges for parents in many areas concerning the everyday lives of young children and adolescents. Excessive screen time, inappropriate content and other risky online behaviors are of a major concern for parents. This makes parental control network devices (e.g., WiFi routers) and software applications a very common solution to help parents in regulating their children's online activities. On the other hand, the majority of such parental control solutions have access to a significant amount of privacy-sensitive data that could be targeted by attackers. Consequently, while these parental control tools should provide peace of mind for parents, they may introduce serious privacy and security issues that can even put children at risk. In this report, we present an experimental framework for systematically evaluating privacy and security issues in parental control software and hardware tools. By utilizing the developed framework, we provide the first comprehensive study of parental control tools on multiple platforms including network devices, Windows applications, and Android apps. Our analysis uncovers pervasive security and privacy vulnerabilities that can lead to leakage of private information, or allow an adversary to fully control the parental control solution and bully, or lure the child. Finally, based on our findings and analysis, we provide some recommendations for developers of these parental control tools.

1 Introduction

Many of today's children cannot imagine their daily lives without without being internet-connected. A recent survey [82] shows that 42% of US children (ages 4 to 14) spend over 30 hours a week on their phones. Nearly 70% of surveyed parents think that such use has a positive effect on their children's development [82]. On the other hand, 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., [30, 5]).

To mitigate such problems and provide a safe, controlled internet experience, many parents and educators at schools 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 [26, 66], indicating a growing adoption of these tools. Such solutions are also recommended by government agencies, such as US FTC [38] and UKCCIS [87]. These solutions offer to help parents maintain control in a wide variety of ways, albeit with somewhat questionable effectiveness/robustness (cf. EU commissioned benchmark at: sipbench.eu), and morality since they, arguably, can act as surveillance tools [96].[1]

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 [56, 67]. Aside

[1]This ethical/moral debate is outside the scope of our work.

from endangering children's safety (online and in the real-world), such leaked children's personal data may be sold by criminals for a price (cf. [98]). Reports, published in the last few years, revealed several privacy and security issues in the analyzed parental control solutions [7, 8, 9, 10, 93]. However, such analysis was performed on a few selected Android apps, and only one network device, even though there are many popular parental control solutions across different platforms (mobile and desktop OSes, and stand-alone/hardware network devices). Note that, unlike other vulnerable products (e.g., buggy gaming apps [94]), or non-complaint products (e.g., see the analysis of free Android apps [75]), which can be removed when such concerns are made public, parental control tools are deemed *essential* by many parents and schools, and thus are not expected to be removed due to the lack of alternatives.

In this work, we undertake the first comprehensive study to analyze different types of parental control hardware/stand-alone and software solutions. We design a set of privacy and security tests, and systematically analyze popular representative parental control solutions available in network devices, Windows and Android OSes. Note that most parental control solutions implement various techniques hindering traffic analysis (e.g., VPNs, SSLpinning and custom protocols). Also, we need to cope with proprietary firmware and code obfuscation techniques when performing static analysis. Understanding long-term behaviors of these solutions by running/using them for days in a realistic way, is also time consuming.

Contributions. Our contributions can be summarized as follows: (i) We develop an experimental framework for systematically evaluating privacy and security issues in parental control software and hardware tools; (ii) We utilize this framework to provide the first comprehensive study of parental control tools on multiple platforms, including 5 network devices, 8 Windows applications, automated analysis (utilizing several existing mechanisms such as Firebase scanner [77], Exodus-Privacy [69], and VirusTotal [91]) for 171 Android apps, and an in-depth comprehensive analysis for 13 of them. The in-depth analysis aims to inspect the apps' web traffic for personally identifiable information (PII) leakage, broken API authentication and potential vulnerabilities. It also identifies third-parties, known trackers, web tracking through HTTP cookies and HTTP Local Storage; (iii) The results of our analysis reveal that the majority of solutions broadly fail to adequately preserve privacy of their users (children and parents).

Disclosure and notable findings. As part of responsible disclosure, we are in the process of contacting the developers and manufacturers of the solutions mentioned in this report; note that some of these solutions have a user base in the millions. We are also sharing our proof-of-concept exploit scenarios and possible fixes with these companies. Our notable findings include:

- The Blocksi parental control router [19] has a critical vulnerability that allows an attacker to remotely inject shell commands by knowing only a parent's email address, enabling an attacker to eavesdrop/modify the home network's traffic or use the device in a botnet (cf. Mirai [14]). Blocksi's firmware update mechanism is also completely vulnerable to a network attacker.

- 8/13 Android apps do not protect their server APIs with authentication, allowing any attacker to view/modify server-stored child information. For example, FamilyTime, which has been certified by kidSAFE [78] (designated by the US FTC as a COPPA Safe Harbor provider [89]), allows anyone to access a child's information by calling an unauthenticated API that requires only a 6-digit sequence-based ID as input.
- 4/13 Android apps allow an attacker to easily compromise the parent account at the server-end, enabling full account control to the child device in some cases (e.g., install/remove apps, allow/block phone calls and internet connections).
- 7/13 Android apps fail to secure backend databases with proper authentication, leading to data exposure of children and parents.
- 6/13 Android apps transmit PII via HTTP. For example, Kidoz, certified by kidSAFE [79], transmits the parental account credentials via HTTP.
- 8/13 Android apps potentially violate US COPPA by failing to collect verifiable parental consent before receiving children's personal information.
- Among the parental control tools with a web interface, all the Android apps and network devices, and 3/4 Windows applications fail to implement HSTS properly, making them susceptible to man-in-the-middle (MITM) attacks via SSLStrip (cf. [54, 52]).
- 3/8 Windows applications utilize a TLS proxy that degrades connection security, by accepting certificates and ciphers that are rejected by leading modern browsers. Another Windows application (Kidswatch) completely lacks HTTPS and communicates with the backend server via HTTP.

2 Related Work

Over the past few years, several parental control mobile apps 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 [56]. Family Orbit exposed nearly 281 GB of children data from an unsecured cloud server [67]. In 2019, a privacy flaw in Kaspersky antivirus and parental control application was found [31]. This application included a script to perform content checking on each page intercepted by a TLS proxy. However, some unique identifiers were also included in the process, allowing an on-path adversary to track the user. In 2010, EchoMetrix settled Federal Trade Commission (FTC) charges for collecting and selling children's information to third-parties through their parental control software [32].

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, including Smart Sheriff [7, 8], Cyber Security Zone [9] and Smart Dream [10]. 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.

Fajardo [6] performed a preliminary analysis of 7 Android apps (few common with us), and reported at least one privacy issue for each. The analysis includes: the features offered by each app, permissions requested vs. required, and leak of information to third-parties.

In 2017, Wisniewski et al. [96] evaluated 42 features offered by 75 parental control Android apps. They showed that most apps value control strategies (e.g., via monitoring/restricting) over self-regulation strategies (e.g., via self-monitoring), and boast the use of privacy invasive techniques rather than a balanced protection mechanism. In 2018, Marsh [57] investigated—via surveys, interviews and user studies—the online risk perception by parents and teens, and measured the effectiveness and usability of two parental control apps. In 2018 Reyes et al. [75] analyzed free Android apps for children for COPPA compliance via an automated framework. 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.

Windows parental control applications have been studied for the security of their TLS proxies [29], but not from a privacy standpoint. Similarly, parental control network devices have not been subject to much analysis, except the Disney Circle, analyzed by Cisco Talos in 2017, and found to host 23 different security vulnerabilities [93]. 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 privacy and security threats in popular parental control solutions across multiple platforms: mobile (Android), desktop (Windows), and stand-alone network devices.

3 Background and Threat Model

We use the term "parental control tools" to cover different types of parental solutions: network devices, Android apps, and Windows applications. 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. Although we do not test these techniques for robustness (cf. sipbench.eu), we note that network devices' web filtering functionality can be easily bypassed via DNS over HTTPS (DoH). This is owing to the fact that DoH protects DNS requests and responses from being inspected or modified by a man-in-the-middle (in this case, the network devices).

Being network-based, parental control devices can only monitor network traffic but cannot inspect the content of encrypted traffic without requiring the installation of companion software on the child's device. 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 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 users connected to it.

Android apps rely on several Android-specific mechanisms, including the following: (1) Device administration [12], which provides device administration features at the system level. Apps may use this feature to control the child's device, gaining access to commands such as lock the device, factory reset, install certificates, and encrypt device storage. (2) Android accessibility service [13], which enables apps to perform several functions including: monitoring user actions by receiving notifications when the user interacts with an app, retrieving window content by inspecting the content of a window the user is interacting with (screen capture), observing typed text by recording text typed by the user (key logging), and control website content by injecting JavaScript code into visited web pages. (3) Mobile device management (MDM) [55], which enables additional control and monitoring features, designed for businesses to fully control/deploy devices in an enterprise setting.[2] (4) Notification access, which 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 [49]), which are used to filter web content. (6) Facebook [33] and YouTube OAuth [40] features, which are used to monitor the child's activities on Facebook (e.g., posts and photos), and YouTube (e.g., playlists and comments).

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: Other user applications are monitored for their usage and duration. (3) User activity monitoring: Some Windows applications take screenshots, record keystrokes, and access the webcam.

3.2 Threat Model

Attacks requiring physical access to either the child/parent device are excluded from our threat model, parental control tools are not designed to protect against this type of threats. We consider the following types of attackers with varying capabilities. (1) On-device attacker: a malicious app with limited permissions on the child/parent device. (2) Local network

[2]Note that, as a mechanism for parental control, MDM features may be just too powerful, and may open doors to abuse. Apple has removed several popular parental control apps from App Store due to their use of such highly invasive features [15]. In contrast, Google Play apparently still allows these features in parental apps.

attacker: an attacker on the same internet-connected local network as the child's 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 the backend server of a parental control solution. (4) Remote attacker: any attacker who can connect to a solution's backend server.

3.3 Potential Security and Privacy Issues

We define the following list of potential privacy and security issues to evaluate parental control tools, which we tested using only our own accounts.

1. *Insecure PII transmission*: PII from client-end is sent to parental control servers without encryption, allowing an adversary to eavesdrop for PII.

2. *PII shared with third-parties*: Explicit transmission of PII to third-parties directly from the client-end.

3. *Insecure mobile backends (Android only)*: The use of misconfigured or unauthenticated mobile backends e.g., Google Firebase [39] by parental apps. Such insecure backends may allow anyone ready access to privacy-sensitive information [16].

4. *Lack of HSTS enforcement*: Misconfigured or unimplemented HSTS (cf. [54, 52]) in HTTPS communications, allowing SSL Stripping attacks. Contained in the server response header, HSTS forces the use of HTTPS for any subsequent connection to a domain for a defined duration.

5. *Lack of authentication*: Blindly accepting queries at the server-end without checking whether the requester owns the account, e.g., logged-in via the parental control interface.

6. *Broken authentication*: Unencrypted storage or transmission of authentication secrets e.g., passwords and session IDs.

7. *Unverified parental consent*: Failure to obtain verifiable parental consent before data collection from children under 13 years old, as mandated by US COPPA [37].

8. *Online password brute-force*: No defense mechanism is implemented to stop unlimited login attempts at the online parental login interface.

9. *Weak password policy*: Acceptance of very weak passwords, e.g., with 4 characters or less.

10. *Unverified password change*: Not requiring knowledge of the original password/PIN, when setting a new password or PIN code for parental accounts.

11. *Uninformed suspicious activities*: Parents not being notified about potentially dangerous activities, e.g., the use of parental account on a new device or password changes in parental accounts.

12. *Faulty setup procedure*: Vulnerable initialization during device setup or account creation (e.g., user registration through HTTP).

13. *Developer SSH access (network devices only)*: The use of SSH backdoors allowing the manufacturer to control individual devices remotely, without parents being aware of such access.

14. *Vulnerable device firmware*: A device firmware or its update mechanism being vulnerable, allowing information disclosure or even full device compromise.

15. *Vulnerable backend*: The use of remotely exploitable outdated software services at the backends.

3.4 Selection of Parental Control Solutions

We chose solutions used in the most popular computing platforms for mobile devices (Android) and personal computers (Windows), and selected network products from popular online marketplaces (Amazon).

We used "Parental Control" as a search term on Amazon and selected five popular products (as apparent from user reviews): KoalaSafe [48], Circle Home Plus [24], KidsWifi [47], Blocksi [19], and Roqos Core [76]. For Windows applications, we relied on rankings and reviews provided by specialized media outlets (e.g. [22, 65, 45]), and selected eight applications. We also checked for the availability of a free trial (see Table 11 in the Appendix). For Android apps, we searched the following terms on Google Play: "Parental Control" and "Family Tracker" between May–Nov. 2019. From a total of 462 apps, we filtered 165 apps with download rates over 10,000. We used this list for automated analysis using Firebase scanner [77] to check for unprotected Firebase databases, Exodus-Privacy [69] to perform static analysis of tracking SDKs, and VirusTotal [91] for malware analysis of the APKs. For in-depth comprehensive analysis, we picked 12 popular apps, most with over 1,000,000 installations. For some apps, we tested multiple versions downloaded from Google Play, and the company's website, along with their paid premium versions (6 apps). We also analyzed the companion app associated with the Circle Home Plus device. Thus our detailed comprehensive analysis focused on 13 apps, including the Circle companion app; see Table 10 in the Appendix. During our analysis of these 13 Android apps, we added 6 APKs downloaded directly from the parental control tool providers' websites that support advanced features, increasing the total number of automatically analyzed APKs to 171.

4 Methodology

We tested each considered parental control tool by mimicking common uses to identify and trigger parental control mechanisms. Once identified, we looked into the potential vulnerabilities in these mechanisms. On Android, we performed basic phone activities (SMS, phone call) and internet-related activities (Instant messaging, internet browsing). On Windows, we automatically generated web browser activities using Selenium [81] to simulate user behavior. For network devices, we manually browsed the internet from protected devices (smartphones, computers). We evaluated the web filtering mechanism by visiting two domains: a university website and an online email service. The first was configured as a blocked domain whenever the parental control tool allowed the user to add blocked domain and the second was used as a control to compare against.

4.1 Experimental Setup

We utilized three experimental setups for each type of parental control tool (see Figures 1, 2, and 3).

1) Network devices test environment. The targeted devices were evaluated in a lab environment simulating the conditions of a domestic network, where they were connected to an internet-enabled router with the OpenWrt firmware [64]. We used a test device that supports web browsing to emulate a child's device. If the parental control device uses ARP spoofing, the test device was connected directly to the router's network. Otherwise, the test device was connected to the parental control device's access point. We captured network traffic on both the test device and the OpenWrt router using Wireshark and tcpdump, respectively. To decrypt TLS traffic generated by the test device, the SSLKEYLOGFILE environment variable was set and Wireshark [95] was configured to read the TLS Pre-Master-Secret from the corresponding file. We used Google Chrome [41] to navigate to different websites on the test device.

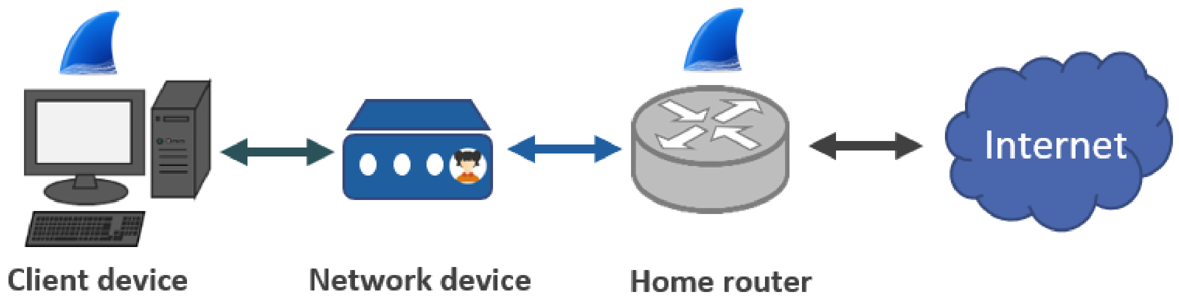


Figure 1: Network devices test environment (Wireshark is installed on both the client device and home router)

2) Windows applications test environment. We tested each Windows applications on a fresh Windows 10 virtual machine with Chrome, Wireshark and Sysinternals pre-installed. We intercepted inbound and outbound traffic using MITMProxy on the host machine and recorded packets using tcpdump.

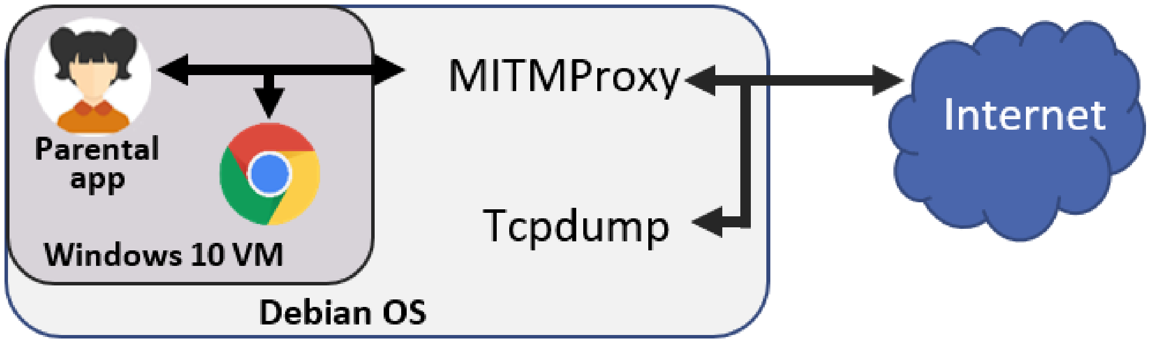


Figure 2: Windows applications test environment

3) Android apps test environment. We maintained two experimental environments to sniff and inspect network traffic originating from the child and parent Android apps. We examined the child apps using a Samsung Galaxy S6 phone running Android 7.0. We also examined all parent apps using a Nexus 4 with Android 5.1.1, except for Life360 which does not support this Android version. Each test environment hosted a virtual Linux environment within the Android OS. We used Debian and MITMProxy [58] on the virtual environment, and configured Android's network settings to proxy all traffic going through the WiFi adapter to an MITMProxy server. We used tcpdump [85] to capture the network traffic. The proxy extracted the shared session keys established with a destination server, enabling us to decrypt HTTPS traffic. Both Android 7.0 and 5.1.1 offer easy support for the virtual Linux environment. Note that although we rooted our test devices to support our test environments, this is not a requirement for an attacker to perform the attacks we describe.

4.2 Network Analysis: Steps and Challenges

To intercept and analyze traffic on our three test environments, we used various tools and techniques. We summarize them here, including the challenges faced.

Network traffic attribution. A key issue is to identify the process that generated the traffic in the absence of the packets' referral metadata. We tested how the parental control tools behave when the child uses her device normally for phone calls, sending messages, browsing the internet and so forth. These activities produce a large amount of traffic that we need to match to the corresponding process. We used the MITMProxy scripting API [58] to read the process information associated with every packet using the `netstat` command.

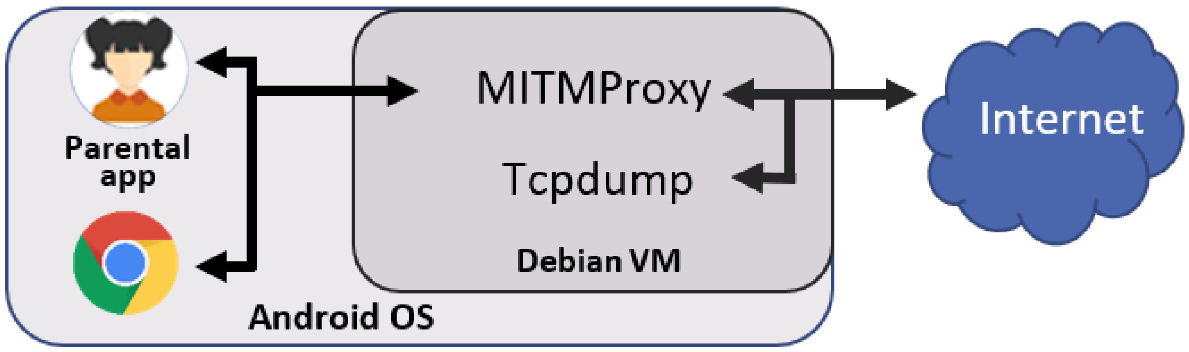


Figure 3: Android apps test environment

To identify third-party domains in the parental control tools traffic, we used the WHOIS [72] registration record to compare the domain owner name to the parental control website owner.
Traffic interception. Most network devices use TLS for communicating with their backends. We could not insert a root certificate on these devices, so some of the network traffic generated by them is completely opaque to us. Instead, we attempted to access the device's shell or its firmware by connecting to a serial port on the device, when possible. To obtain a serial interface, the device's casing was opened and the PCB inside was inspected for pins labelled as UART. If found, we connected a USB-to-TTL Adapter (FT232RL) to these pins and used PuTTY [71] as a serial client. To automatically find the correct baudrate for the serial connection, the baudrate.py tool [43] was used. We managed to obtain shell access to the KoalaSafe and Blocksi devices. Once the serial connection is established, the device is rebooted to examine its boot sequence, showing that they are running an OpenWrt firmware image. Although KoalaSafe and Blocksi shells are password-protected, we entered failsafe mode, a feature of OpenWrt, during boot, which allowed us to bypass authentication. With shell access, we attempted to understand how these network devices communicate over the network.

As for Android apps, starting from Android 7.0, apps no longer trust user certificates by default. Therefore, we installed the MITMProxy root certificate in Android's system trusted store, which allowed us to intercept TLS traffic from Android apps. In cases where an app uses certificate pinning to refuse server certificates signed by any CA other than the one with the pinned certificate in the app, we used SSLUnpinning [2] to attach several hooks in the SSL classes in order to bypass this feature and intercept the communication. Some apps install a VPN on the child device to filter and block websites. We intercepted traffic by deleting the VPN configuration from Android setting on the child device. This was not possible in one case where the child app would stop working if the VPN is deleted. In this case, we manually updated the app configuration file to disable the setup of the VPN on startup of the app on the child device.

All the Windows applications tested, except one, used the Windows trusted root certificate store. For the ones that do, we installed the MITMProxy root certificate and inspected the content of the network traffic. One application used its own encrypted certificate store. Two applications used a custom protocol to communicate with their server, hindering our analysis.

4.3 Additional Analysis

Vulnerability scanning tools. To complement the network information-gathering phase, we used automated vulnerability scanning software. Network devices operate as routers and thus have network services listening on open ports. An outdated and vulnerable version of such a service could lead to a possible compromise of the device. Therefore, we scanned the network devices with OpenVas [63], Nmap [62], Nikto [25] and Routersploit [74], and matched the identified versions against vulnerability databases.

TLS vulnerability analysis. To assess the quality of TLS configuration in the parental control tools, we first set up a WiFi access point [28] with MITMProxy, SSLStrip2 [50] and Wireshark [95] installed. Then, we conducted a comprehensive audit of the different components of the parental control tools, including child apps, parent apps, associated websites, and Windows applications.

Firebase analysis. Most analyzed Android apps use Google Firebase as a backend service. Thus, we analyzed the app's Firebase configuration for security issues. We performed an automated analysis on all 171 Android apps using Firebase Scanner [77], a tool for reverse engineering and detecting security misconfigurations in Firebase. Critical misconfigurations can allow attackers to retrieve all the unprotected data stored on the cloud server.

Privacy leakage. We examined the collected traffic to check for PII transmitted in plaintext. This task was automated by parsing and committing the collected Wireshark web traffic files to a central SQLite database. We also examined the collected traffic to check for leakage of PII to third-party domains.

Network devices code analysis. For the network devices accessed through the serial interface, we conducted an analysis of the services running on the device. These services include the application logic used by the device in its domain filtering mechanism. We manually analyzed these services' code to identify the requests sent by the device to their backend APIs and their expected responses. Thus, without decrypting the network traffic, we could identify information being sent to the backend. Aside from API communication, the application logic and update mechanisms were also reviewed for vulnerabilities or insecure practices such as unsanitized input, or updating through HTTP.

Password brute-force attack. We used the Burp Suite Intruder tool [68] to perform password brute-force attacks on our own accounts.

API authentication. The parameters used for API authentication, such as API keys, tokens or unique identifiers, were retrieved from the API requests. These parameters were then assessed in terms of their predictability and confidentiality. For instance, a device using its own MAC address for API authentication is deemed insecure as an attacker can easily find valid MAC addresses by eavesdropping on the network traffic.

Backend vulnerabilities. Due to ethical/legal concerns, we refrained from using any invasive vulnerability scanning tools to assess backend servers. Instead, we looked into the information disclosed by web servers or frameworks in their HTTP headers "Server" and "X-Powered-By." The software components running on the backend server were extracted from these headers. Disclosed software components were matched against the CVE database to detect known vulnerabilities associated with these versions.

5 Results

In this section, we report our findings on the tested privacy and security issues (as outlined in Section 3.3). Note that the dynamic analysis of all the tested solutions produced over 50GB of network traffic. We automatically analyzed the traffic for PII leakage, broken API authentication, potential vulnerabilities, and tracking measurements.

5.1 Network Devices

This section details the results of analyzing the five network devices. We purchased and manually analyzed these devices between Mar. 2019 to Feb. 2020. Overall, we found critical vulnerabilities in 3/5 devices.

5.1.1 Device Vulnerabilities

Insecure update mechanism. The importance of securing the update mechanism has been known for years, cf. [18]. 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 tested this vulnerability and confirmed it to be exploitable.
Remote command injection. Remote command injection occurs when unsanitized user input is passed to a system shell for execution. We found an exploit that enables executing a command as root on the Blocksi device. An attacker can issue system commands as if she were at a terminal on the router. We tested this vulnerability and confirmed it to be exploitable. The vulnerability can be exploited by sending a `router_setGeneralSettings` request to the Blocksi API, 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 and updates its local configuration. 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.
Outdated SSH server. KoalaSafe is running the Dropbear v2014.63 SSH server/client (released on Feb. 19, 2014), which has 4 known remote code execution vulnerabilities. Therefore, a remote attacker can exploit these vulnerabilities and compromise the device if a reverse tunnel is opened as discussed in Section 5.1.3.

5.1.2 Information Leakage and Weak Authentication

Device ID enumeration. We found an API call in the KoalaSafe device that enables an attacker to enumerate all MAC addresses of every active device. This allows a remote attacker if combined with other potential exploits (e.g., on the SSH server) to compromise the network device. This API call usually runs on a daily basis to potentially create an SSH reverse tunnel to the KoalaSafe device. The API response varies based on whether the requested device MAC exists or not. If the network device exists, the response body is a valid integer. otherwise it is the string "None". Moreover, the KoalaSafe MAC address space is relatively small because KoalaSafe uses the GuangLia network interface, which reserves a range of 1,048,576 potential MAC addresses [83].
MAC address in DNS queries. We found that the KoalaSafe and Blocksi network devices append the child'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 [27].
Weak backend API authentication. The KoalaSafe API authentication works by sending multiple elements, a device-generated authentication token, the device's date and time, and the device's MAC address to the login endpoint at: `https://api.koalasafe.com/api/auth/device/login`. The server then sends back an HTTP status code indicating success or failure, and a master session ID cookie. However, the authentication token and device time required can be obtained by visiting `https://device.koalasafe.com/auth.lua`. The MAC address can be obtained via `https://device.koalasafe.com/status.lua`. Thus, a local network attacker can easily collect the information needed for authentication and execute any API request.

As for the Blocksi API, the API uses a combination of the device's serial number and the registered user's email 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 serial number (SN) from `https://service.block.si/config_router_v2/router_checkRouters/null/{SN}`, or vice-versa from `https://service.block.si/config_router_v2/router_checkRouters/{email}`. By using both parameters in an API request, any remote attacker can authenticate to the server, which holds sensitive data about the home network (e.g., the WiFi password, MAC addresses of connected devices).
Weak device API authentication. The Circle Home Plus device locally stores the users' profile information along with device information. Upon reception of a GET request, the Circle Home Plus identifies the child's device based on the requester's MAC address. It then responds with the associated child account usage history and statistics, profile ID and information, and Circle-specific information. Moreover, the device contains information about all configured profiles on the account, including their age groups and usernames. It also stores a photo for each profile if provided by the user. This profile photo could be fetched by providing its profile ID via: `http://10.123.234.1/api/USERPHOTO?profileID=[profileID]`. Both API responses are sent in plaintext over the local network whenever a child device attempts to access a restricted domain. Thus, an eavesdropper, such as a malicious house guest, or an attacker who cracked the WiFi password, can easily intercept and steal

sensitive information being exchanged over the network. Alternatively, a local attacker who can spoof her MAC address can easily extract this information. A malicious app or SDK installed on the child's device can also launch this attack, which tested by creating a dummy app and collecting data in the background.

5.1.3 Insecure Practices

SSH reverse tunnel. Through the static analysis of the device, we found that one KoalaSafe script "phonehome.sh" queries its backend daily via a particular API. If the value retrieved corresponds to a valid port number, the KoalaSafe device creates an SSH reverse tunnel on the specified port number on `et.koalasafe.com`. When this tunnel is created, SSH clients from outside the network can connect to the device. Although SSH authentication is still required, an attacker can exploit the SSH server as mentioned in Section 5.1.1.

Table 1: Vulnerable software components on backend APIs.
N/A: Not enough information could be extracted.

Device	Software Components	# of CVEs
KoalaSafe	Apache 2.4.34	11
Circle	Nginx 1.15.5	3
KidsWifi	Nginx 1.10.2	3
	PHP 7.0.27	26
Blocksi	OpenResty, Google Frontend	N/A
Roqos	N/A	N/A

Outdated backend servers. The results of our analysis of the backend server APIs is summarized in Table 1. Of the subdomains contacted by the Circle device, only one, `urldb.meetcircle-blue.co`, incorporates a "Server" HTTP header in its response. The other subdomains were opaque in terms of the web framework they are running. Based on the Server header, `urldb.meetcircle-blue.co` runs version 1.15.5 of the nginx web server, with 3 known CVEs, all related to the server's availability. The Blocksi's API backend only indicates that it runs on OpenResty and Google Frontend. It does not report the versions of these components. As for the Roqos, the backend servers do not disclose the web server or framework they are running in their response headers.
Faulty setup procedure. During the setup procedure of KidsWifi, the device creates an open wireless Access Point (AP) called "set up kidswifi". The user must use this AP's captive portal to configure the KidsWifi device to connect it to their home network. However, as this AP is not password protected and the client-device communication happens through HTTP, the home router's WAN and KidsWifi's LAN credentials become available to a local attacker.
Insecure storage on web interface. The KidsWifi user web interface generates a token when a user connects to it. This token is used as an authentication means to access personal information via the device's APIs. The token generated is stored in plaintext on the local

storage, which is not designed to store sensitive information [35]. Local storage does not have the cookies' "http-only" flag protection. Thus, information stored in local storage can be requested by any JavaScript contained in the page, including imported libraries. To obtain the token, an attacker could leverage cross-site scripting (XSS) attacks. Moreover, this token is the only authentication method used to control API calls and it is unique per device. This design choice also disallows the removal of a (malicious) parent account; any user who ever had access to the interface can just save the token and keep requesting sensitive information.

5.2 Android Apps

We performed an automated analysis of 171 parental control Android apps to detect vulnerable backend databases and statically analyzed the apps to check their use of tracking SDKs. We then perform a comprehensive privacy and security analysis of 13 widely used parental-control Android apps (six paid and seven free versions). We analyzed each app between Mar. 2019 to Feb. 2020. In this section, we present the results of our analysis. Overall, we found 79 significant vulnerabilities across these 13 apps. Table 2 summarizes these vulnerabilities where apps are ordered alphabetically. All but one app (Circle), contain at least one major vulnerability that can harm the child's privacy.

5.2.1 Data Exposure

Insecure cloud backend database. We followed a similar approach to Appthority's work [16] on scanning apps for Firebase misconfigurations. We run the Firebase scanner on 171 Android apps and found seven 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. We created a parental account on the seven apps, and then we updated the Firebase scanner to automatically search for information from our configured parent account. We found three apps that expose personal information: 1) FamiSafe with 500,000+ installs exposes the parent account email; 2) Locate : Family Locator - GPS Tracker [23] with 10,000+ users exposes the child name, child phone number, and child email; and 3) My Family Online "ailemonline" [84] with 10,000+ users exposes the child name, child phone number, parent mobile number, parent email, and apps installed on child phone.

Lack of HSTS enforcement. We found 9/13 Android apps that enable SSLStrip attacks. We connected the device with the Android app to our WiFi access point with MITMProxy, SSLStrip2 installed. Wireshark was 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 looking for signs of successfully SSL Stripping attack on the traffic. We confirmed the success of the attack by comparing the result to the corresponding traffic in normal testing environment (i.e., without the SSLStrip). We found that nine Android apps that transmit 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 different device.

In Kidoz, we were able to see the parent's credit card account number and email in HTTP when using their BlueSnap online payment solution [20], while connected to our WiFi access point. This was possible because the online payment server is not configured to use HSTS. In Qustodio, we were able to extract the child Facebook credentials provided by the parent during the configuration of the monitoring component (see Figure 4(b) in the Appendix). 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.

Lack of authentication. We found 8/13 Android apps lack authentication for accessing PII. Prominent examples include the following: In FamilyTime, a six digit parameter *childID* is generated through a sequential counter that is incremented by one per user. An attacker can retrieve the child name, gender, data of birth, email address, and child phone for a child through an API that requires only the *childID* value. Hence, an attacker can remotely exploit this vulnerability at a large scale, e.g., using:

```
$ curl -v https://mesh.familytime.io/v2/child/Android/profile/456***
```

In FamiSafe, an attacker can retrieve all the child social media messages and YouTube activities labeled as suspicious through an API. This API requires *deviceid*, *memberid*, *client_sign*, and *access token* parameters. Any app installed on the child device can access the required parameters from the FamiSafe log file on the shared external storage. An exploitation example is shown in the code snippet below.

```
$ curl -v https://u.famisafe.com/load-page/index?page=suspicious-text/deta
il&access_from=1&device_id=165***&member_id=1045***&client_sign={fffff*
**-be**-19ec-0000-000075b3****}&access_token=dtwMtFarI********&lang=en
```

Broken authentication. We found 4/13 Android apps where an adversary can easily compromise the parental accounts. For instance, in SecureTeen, we found a link that can be used to authenticate the user to the parental control account. This link enables any adversary to remotely compromise any parental account by knowing only the parent's email. After that, the adversary obtains full access to the parent account, including the ability to monitor and control the child device. An exploitation example is as follows.

```
https://cp.secureteen.com/auth.php?&productName=secureteen&resellerId=
careteen&page=menu&loginFromApp=Yes&j_username=parentemail**@gmail.com
&gType=monitoring
```

Kidoz transmits the user email and password via HTTP when the "Parental Login" link is clicked from kidoz.net home page. KidsPlace and Qustodio leak session authentication cookies via HTTP. The KidsPlace authentication cookie validity is one year while the Qustodio session cookie is valid for two hours. Although the session key expiry is short for Qustodio, this period might still be enough to enable the attacker to access sensitive information about the child including, the child's current location, and history of movements. The attacker can also remotely control functions in the child's 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.

Insecure storage of sensitive data. We found 3/13 Android apps that do not encrypt stored user data on shared external storage that can be accessed by any other apps with the permission to access SD card. Examples of the sensitive information include the child's messages and social media chats, visited websites, and even authentication tokens that we were able to use to read private information from the child account remotely. KidsPlace stores sensitive information on the shared external storage including the parent's email and the PIN code. Life360 stores a log file with sensitive information (e.g., parents and children emails, phone numbers, and geolocation data) on the shared storage.

5.2.2 PII Collection, Sharing, and Leakage

PII data collection. We summarize the personal data collected by the child and parent apps in Tables 3, 4 and 5. The MobileFence app is initially setup by default to monitor both the child and parent devices. The SecureTeen app reads all the child communications including the SMS and call history (that are stored on the device and occurred even before the installation of the app). SecureTeen also records all social media activities and keystrokes on the child device through keyloggers, and it also takes screenshot from the child device. SecureTeen accesses all the pictures stored on the shared external storage, including Whatsapp, Snapchat, Instagram, and Viber pictures. The app uploads all the pictures to a server (cp.secureteen.com) in the USA.

Insecure PII transmission. We found 6/13 Android apps that transmit PII via HTTP. FindMyKids transmits the child's surrounding sounds and the child's picture. KidsControl transmits the parent's name and email, geolocation, and SOS requests. MMGuardian transmits the parent's email and phone number, and child's geolocation. MMGuardian and SecureTeen transmit the child visited URL (encoded in Base64) to a third-party domain classifier `Komodia.com` [49]) via HTTP. MobileFence transmits the child's profile picture via HTTP.

Share PII with third-parties. We found that 12/13 Android apps share personal and unique device information with third-party domains (see Table 6). Prominent examples include: FamiSafe which detects any suspicious photo saved on the child device and stores the photo on Alibaba cloud in the USA [4]. FindMyKids shares child and parent Android IDs with a third-party domain in Russia (`yandex.net`). The parameter that restricts tracking advertising on the device was not enabled (i.e., limit_ad_tracking parameter is set to zero) [97]. ScreenTime shares the child Android ID with facebook.com.

5.2.3 Unverified Parental Consent

Only 5/13 Android apps comply with the US COPPA parental verifiable consent policy; apparently, four of them comply with this guideline and Life360 partially complies. For Life360, when adding a child younger than 13, the app requires the parent to sign and return a consent form via postal mail or email (scanned). However, the child age is not mandatory to create a child account in the app. Therefore, the verifiable consent can be bypassed. Life360 may sell personal information to third-parties as they state explicitly in their privacy policies [53]. Moreover, to comply with California Consumer Privacy Act (CCPA), Life360 allows the user to opt-out of selling her personal information to third-parties. Again, the child age is not mandatory to create a child account.

COPPA allows retention of children data for only as long as is reasonably necessary to fulfill the purpose for which the information was collected [37]. However, we found that 6/13 Android apps store child information in their database for over six months after the expiry of the account trial period. For example, FamilyTime keeps the child name, gender, DOB, email, phone number, movement history, contacts, installed app, time zone, and WiFi name. KidsControl keeps the child name, child picture, and last known location.

5.3 Windows Applications

We collected network traffic by running the Windows applications for several days to understand their long-term behaviors, summarized in Tables 8 and 9. All the tested Windows applications relied on TLS interception to operate except Kidswatch. However, we noticed weaknesses in the certificate validation carried out by the proxy in several applications. Qustodio and Dr. Web proxies accepted intermediate certificates signed with SHA1, despite the enhanced collision attack on SHA1 [51]. Dr. Web's proxy accepted communication using Diffie-Hellman 1024 despite it being considered breakable [3] and deprecated in Safari and Chrome since 2016 (Chrome 53) [1]. In addition, none of the Windows applications' proxy rejected revoked certificates. We also found that upon uninstallation of the Windows 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.

Furthermore, we found that some Windows applications' servers also do not always 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. We found that Qustodio, Spyrix, and KidLogger's online user interface were vulnerable to SSL stripping due to an Lack of HSTS enforcement.

Interestingly, the analyzed anti-virus applications with parental control features (Kaspersky, Dr. Web, and Norton) use apparently non-standard encryption/encoding to communicate with their backend. 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 will not block this action nor warn the user.

Kidswatch is no longer in the top ranking of the years 2018-2019 but remains advertised by some sites, and was part of the top ranking until 2017. However, this application was found to be defective on several levels. Among the analyzed applications, it is the only one that sends unencrypted traffic to its server.

6 Identifying Third-parties

We found explicit data sharing between Android apps and third-parties (not in Windows/network solutions).

Static analysis. We use Exodus-Privacy [70] to extract the list of third-party SDKs from all 171 APKs; see Table 12 in the Appendix. 15 APKs have 9–22 tracking SDKs (see Table 14 in the Appendix).

Dynamic analysis. To verify existence of the third-parties from our static analysis, we also analyzed the web traffic generated by the 13 apps on the child device; see Table 13 in the Appendix. We also collected and analyzed the persistent tracking HTTP cookies stored in the child and parent apps. For example, we found FamiSafe creates HTTP cookies on the child device for domains google.com and youtube.com. KidsControl creates HTTP cookies on child device for domain openstreetmap.org. It also creates HTTP cookies on parent device for third-party domains such as doubleclick.net, yandex.ru, and openstreetmap.org. ScreenTime creates HTTP cookies on parent device for domains facebook.com, `strip.com`, and google.com. For apps that have a safe browser shipped with their Android apps, such as Kidoz, MMGuardian, and KidsPlace, we found that all safe browsers allow saving tracking HTTP cookies and trackers on HTTP Local Storage [92] on the child device (e.g., HTTP Local Storage is used by the Optimizely.com tracker).

We found that one of the network devices' companion app, Circle, includes a third-party analytical SDK. This SDK was developed by Kochava, a company specialized in mobile app analytics. When the app is launched or returns to the foreground, information is shared with Kochava, including Device ID (enables tracking across apps), device data (enables device fingerprinting for persistent tracking). It is worth noting that Disney, a former partner of Circle, is the target of a class action lawsuit for using a similar SDK in child-intended apps [88].

7 Potential Practical Attacks

In this section, we briefly discuss the impact of exploiting some of the discovered vulnerabilities of the examined parental control tools.

Device compromise. Device compromise presents serious security and privacy risks, especially if a vulnerability can be exploited by a remote/network attacker. 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 the two vulnerabilities to take control over the Blocksi device by simply knowing the parent's email address. An adversary can remotely leverage the APIs described in Section 5.1.2 and retrieve the device serial number using the parent's email. With both the serial number and the email, the attacker can access a broad range of APIs. In particular, using the serial number and email, an attacker can exploit the command injection vulnerability as discussed in Section 5.1.1 and spawn a reverse TCP shell on the device. At this stage, the attacker gains full control of the device and can eavesdrop and modify unencrypted network traffic, disrupt the router's operation (cf. DHCP starvation [86]), or use the device as part of a botnet (cf. Mirai [14]).

Account takeover. Parental accounts can be compromised in multiple ways. First, none of the parental control tools' web user interface except Roqos enforced HSTS, and almost all the interfaces were found vulnerable to SSL Stripping attacks. Therefore, an on-path attacker can possibly gain access to the parent account using SSLStrip, unless parents carefully check the URLs before submitting credential information. Second, login pages that allow unlimited number of password trials could allow an adversary to brute force passwords, particularly if they are weak. Note that most parental control tools' password policies are apparently weak considering the NIST guideline [42]. Some products accept passwords as short as one character. Third, when system suffers from broken authentication, it allows a remote attacker to access the parental account without credentials. In SecureTeen we found an API, described in Section 5.2.1, that can authenticate the user to the parental control account. This API enables any adversary to remotely compromise any parental account by knowing only the parent email address. When logged-in, the attacker has access to a large amount of PII (see Figure 4(a) in the Appendix), including web history, social media messages and pictures. More critically, the app tracks the child location, and reads SMS messages and phone history. Thus, this vulnerable app may present dire consequences to the children. A physical attacker can take advantage of the tracker information to locate the child's device, and possibly even harm the child.

Data leakage from backends. Failure to protect the parental control backend databases exposes sensitive data about both the parent and child at a large scale. Firebase misconfigurations exposed data that belongs to 500K+ children and parents from three parental control Android apps. We uncovered sensitive data exposures from those vulnerable backend databases, including parent name, parent email, child name, child email, child phone number and apps installed on child phone (verified using our own account information). Anyone, with malicious intent or otherwise, can easily access such information, which may lead to potential exploitation of children both online and offline.

PII on the network. COPPA requires that companies must adopt reasonable security procedures to protect children's information [37]. However, we found several parental control tools transmitting PII via insecure communication channels. 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 voice records and her surrounding, and also recognize the child from her photo. Another Android app allows the child to send SOS messages when she is in a dangerous situation. However, the SOS requests are sent via HTTP. An attacker can identify and drop the SOS message, leaving child isolated. Moreover, KoalaSafe and Blocksi network devices append the child's device MAC address to the outgoing DNS requests. This allows malicious adversaries to track the family's online web activities.

8 Recommendations

In this section, 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. Moreover, they should have a process to address vulnerabilities such as responsible disclosure or bug bounty programs. Currently, none except Kaspersky participate in such programs.

Enforcing best practices. Parental control companies should rely on publicly available guidelines and best practices, including proper API authentication and web security standards [35, 36]. 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 [90]. In the case of network devices, manufacturers should employ a secure firmware update architecture (see e.g., IETF [61]). 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 tool should not store PII which is not required for the tool'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 [44].

Limiting third-parties and SDKs. Parental control tools should limit the usage of trackers and tracking SDKs in apps intended for children. For example, Branch prohibits the use of their SDKs in any apps or websites that are directed to children under 13 [21]. This is to limit the data collection from children to comply with COPPA.

9 Conclusion

Parental control solutions are used by parents to help them protect their children from online risks. Some of these solutions have made news in the recent years for the wrong reasons. Our cross-platform comprehensive analysis of popular solutions (including 5 network devices, 8 Windows applications and 13 Android apps) shows systematic problems in the design and deployment of *all* these solutions (except Roqos) from a privacy and security 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.

References

[1] Remove DHE-based ciphers. `https://www.chromestatus.com/feature/5128908798164992`.

[2] ACPM. SSLUnpinning - Xposed Module. `https://github.com/ac-pm/SSLUnpinning_Xposed/`.

[3] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[4] Alibaba Cloud. Alibaba cloud regions and endpoints. `https://www.alibabacloud.com/help/doc-detail/31837.html`.

[5] K. Allison. *Online Risks, Sexual Behaviors, And Mobile Technology Use In Early Adolescent Children: Parental Awareness, Protective Practices, And Mediation*. PhD thesis, University of South Carolina, 2018. `https://scholarcommons.sc.edu/etd/4950/`.

[6] Álvaro Feal Fajardo. Study on privacy of parental control mobile applications. Master's thesis, Universidad Politécnica de Madrid, IMDEA Software Institute, 2017. `http://oa.upm.es/47135/1/TFM_ALVARO_FEAL_FAJARDO.pdf`.

[7] C. Anderson, M. Crete-Nishihata, C. Dehghanpoor, R. Deibert, S. McKune, D. Ottenheimer, and J. Scott-Railton. Are the Kids Alright? Digital Risks to Minors from South Korea's Smart Sheriff Application, 2015. Article (Sept. 10, 2015) `https://citizenlab.ca/2015/09/digital-risks-south-korea-smart-sheriff`.

[8] C. Anderson, M. Crete-Nishihata, C. Dehghanpoor, R. Deibert, S. McKune, D. Ottenheimer, and J. Scott-Railton. The Kids are Still at Risk Update to Citizen Lab's "Are the Kids Alright?" Smart Sheriff report, 2015. Article (Nov. 01, 2015) `https://citizenlab.ca/2015/09/digital-risks-south-korea-smart-sheriff`.

[9] C. Anderson, M. Crete-Nishihata, C. Dehghanpoor, R. Deibert, S. McKune, D. Ottenheimer, and J. Scott-Railton. Safer without Korean child monitoring and filtering apps, 2017. Article (Sept. 11, 2017) `https://citizenlab.ca/2015/09/digital-risks-south-korea-smart-sheriff`.

[10] C. Anderson, M. Crete-Nishihata, C. Dehghanpoor, R. Deibert, S. McKune, D. Ottenheimer, and J. Scott-Railton. Still safer without another look at Korean child monitoring and filtering apps, 2017. Article (Nov. 27, 2017) https://citizenlab.ca/2015/09/digital-risks-south-korea-smart-sheriff.

[11] Android. Android activity recognition permission. https://developer.android.com/guide/topics/location/transitions/.

[12] Android. Android device administration. https://developer.android.com/guide/topics/admin/device-admin/.

[13] Android. UI/Application Exerciser Monkey. https://developer.android.com/studio/test/monkey.html.

[14] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the Mirai botnet. In *USENIX Security Symposium (USENIX Security'17)*, Vancouver, BC, Canada, 2017.

[15] Apple. The facts about parental control apps, 2019. News article (April 28, 2019) https://www.apple.com/ca/newsroom/2019/04/the-facts-about-parental-control-apps/.

[16] Appthority. Appthoritiy: Enterprise mobile threat report - firebase vulnerability: Exposing sensitive data via thousands of mobile apps.

[17] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[18] A. Bellissimo, J. Burgess, and K. Fu. Secure software updates: Disappointments and new challenges. In *USENIX Workshop on Hot Topics in Security (HotSec'06)*, Vancouver, BC, Canada, 2006.

[19] Blocksi Inc. Blocksi parental control router. https://blocksi.net/parental-control.php.

[20] Bluesnap.com. Bluesnap: Online payment solutions. https://home.bluesnap.com/.

[21] Branch Metrics, Inc. Terms & policies. https://branch.io/policies/.

[22] C. Marshall and C. Ellis. The best free parental control software 2019. https://www.techradar.com/news/the-best-free-parental-control-software/.

[23] Care Apps Studio. Locate : Family Locator - GPS Tracker. https://play.google.com/store/apps/details?id=com.careapps.locate&hl=en_CA.

[24] Circle Media Labs Inc. Circle Home Plus Parental Control Device. https://meetcircle.com/.

[25] CIRT.net. Nikto web server scanner. https://cirt.net/Nikto2/.

[26] Common Sense Media and SurveyMonkey. Think you know what your kids are doing online? think again. Survey report (Dec. 11, 2017), https://www.commonsensemedia.org/blog/think-you-know-what-your-kids-are-doing-online-think-again.

[27] M. Cunche. I know your MAC address: targeted tracking of individual using Wi-Fi. *Journal of Computer Virology and Hacking Techniques*, 10:219–227, 2014.

[28] David Schütz. A python program to create a fake AP and sniff data. https://github.com/xdavidhu/mitmAP/.

[29] X. de Carné de Carnavalet and M. Mannan. Killed by proxy: Analyzing client-end tls interception software. In *Network and Distributed System Security Symposium*, 2016.

[30] DQinstitute.org. Nearly two-thirds of children surveyed around the world are exposed to cyber risks, first-ever global child online safety index reveals. Online article (Feb. 11, 2020). 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/.

[31] R. Eikenberg. Kaspersky script injection. https://www.heise.de/ct/artikel/Kasper-Spy-Kaspersky-Anti-Virus-puts-users-at-risk-4496138.html.

[32] EPIC.org. FTC settles with company that failed to tell parents that children's information would be disclosed to marketers, 2010. https://www.ftc.gov/news-events/press-releases/2010/11/ftc-settles-company-failed-tell-parents-childrens-information.

[33] Facebook.com. Manually Build a Login Flow. https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow/.

[34] Familytime.io. Family Time full app with Calls and SMS monitoring. https://download.familytime.io.

[35] O. Foundation. HTML5 security cheat sheet. https://owasp.org/www-project-cheat-sheets/cheatsheets/HTML5_Security_Cheat_Sheet.

[36] O. Foundation. REST security cheat sheet. https://owasp.org/www-project-cheat-sheets/cheatsheets/REST_Security_Cheat_Sheet.

[37] Ftc.gov. 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.

[38] FTC.gov. Parental controls. Online article. https://www.consumer.ftc.gov/articles/0029-parental-controls.

[39] Google. Google Firebase. https://firebase.google.com/.

[40] Google. Implementing OAuth 2.0 Authorization. https://developers.google.com/youtube/v3/guides/authentication.

[41] google.com. Google chrome. https://www.google.com/chrome/.

[42] P. A. Grassi, R. A. Perlner, E. M. Newton, A. R. Regenscheid, W. E. Burr, J. P. Richer, N. B. Lefkovitz, J. M. Danker, and M. F. Theofanos. Digital identity guidelines: Authentication and lifecycle management [including updates as of 12-01-2017]. Technical report, 2017.

[43] Hefner, Craig. Python package for finding the baud rate of serial connections. https://github.com/biw/Baudrate.py.

[44] IETF.org. HTTP Strict Transport Security (HSTS). https://tools.ietf.org/html/rfc6797.

[45] Jon Martindale. Keep your kids safe online with these great parental control tools. Article (Dec. 11, 2019). https://www.digitaltrends.com/computing/best-free-parental-control-software/.

[46] Kidoz.net. Kidoz app downloaded from website offers additional features. https://kidoz.net/parents-corner/.

[47] Kids Wireless Inc. KidsWifi. http://kidswifi.com/.

[48] KoalaSafe Inc. KoalaSafe Family Friendly Wireless Router with Parental Controls/Access. https://koalasafe.com/.

[49] Komodia.com. Komodia URLs classification SDK. https://url-classification.io/industry/parental-control/.

[50] L. Nve. SSLStrip2. https://github.com/LeonardoNve/sslstrip2/.

[51] G. Leurent and T. Peyrin. From collisions to chosen-prefix collisions application to full SHA-1. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 527–555. Springer, 2019.

[52] X. Li, C. Wu, S. Ji, Q. Gu, and R. Beyah. HSTS measurement and an enhanced stripping attack against HTTPS. In *International Conference on Security and Privacy in Communication Systems*, pages 489–509. Springer, 2017.

[53] Life360. Life360 Privacy Policy. https://life360.helpshift.com/a/life360-family-locator/?s=privacy-policy-tos&f=full-privacy-policy&l=en&p=web.

[54] M. Luo, P. Laperdrix, N. Honarmand, and N. Nikiforakis. Time does not heal all wounds: A longitudinal analysis of security-mechanism support in mobile browsers. In *Network and Distributed Systems Security (NDSS'19)*, San Diego, CA, USA, 2019.

[55] J. Madden and B. Madden. *Enterprise Mobility Management: Everything you need to know about MDM, MAM, and BYOD.* Jack Madden, 2013.

[56] Mark Jones, Komando.com. Parental control app database exposed, leaving kids' information compromised, 2018. News article (May 21, 2018) `https://www.komando.com/happening-now/461381/parental-control-app-database-exposed-leaving-kids-information-compromised`.

[57] A. Marsh. *An Examination of Parenting Strategies for Children's Online Safety.* PhD thesis, Carnegie Mellon University, 2018.

[58] MITMProxy.org. HTTPS proxy. `https://MITMProxy.org/`.

[59] MMGuardian.com. MMGuardian app downloaded from website (Normal Download). `https://family.mmguardian.com/getLatestKidApp/`.

[60] Mobile Fence Support Team. Mobile Fence Plugin to monitor calls and SMS logs. `https://bit.ly/2EYQjfV`.

[61] B. Moran, H. Tschofenig, D. Brown, and M. Meriac. A Firmware Update Architecture for Internet of Things. Internet-Draft draft-ietf-suit-architecture-08, Internet Engineering Task Force, Nov. 2019. Work in Progress.

[62] Nmap.org. Nmap network mapper. `https://nmap.org/`.

[63] OpenVas.org. OpenVas open vulnerability assessment scanner. `https://www.openvas.org/`.

[64] OpenWrt Project. OpenWrt. `https://openwrt.org/`.

[65] PCMag.com. The best parental control software for 2019. `https://www.pcmag.com/article2/0,2817,2346997,00.asp/`.

[66] Pew Research Center. Parents, teens and digital monitoring. Survey report (Jan. 7, 2016), `http://www.pewinternet.org/2016/01/07/parents-teens-and-digital-monitoring/`.

[67] Pierluigi Paganini. Parental control spyware app family orbit hacked, pictures of hundreds of monitored children were exposed, 2018. News article (Spet. 4, 2018) `https://securityaffairs.co/wordpress/75888/data-breach/family-orbit-hacked.html`.

[68] Portswigger.net. The burp suite family. `https://portswigger.net/burp`.

[69] E. Privacy. Platform to audit trackers used by android application. https://github.com/exodus-privacy/exodus.

[70] E. Privacy. The privacy audit platform for android applications. https://reports.exodus-privacy.eu.org/en/trackers/.

[71] PuTTY Project. SSH and Telnet client. https://www.putty.org/.

[72] Pypi.org. Python WHOIS library. https://pypi.org/project/whois/.

[73] Qustodio.com. Download of Qustodio app with Calls and SMS monitoring feature. https://www.qustodio.com/en/downloads/android-with-calls-and-sms-download/.

[74] Reverse Shell Security. Routersploit embedded devices exploitation framework. https://github.com/threat9/routersploit/.

[75] I. Reyes, P. Wijesekera, J. Reardon, A. E. B. On, A. Razaghpanah, N. Vallina-Rodriguez, and S. Egelman. "Won't somebody think of the children?" examining COPPA compliance at scale. *Proceedings on Privacy Enhancing Technologies*, 2018(3):63–83, 2018.

[76] Roqos. Roqos Core. https://www.roqos.com/.

[77] S. Sahni. Firebase scanner. https://github.com/shivsahni/FireBaseScanner.

[78] Samet Privacy, LLC. Official membership page. https://www.kidsafeseal.com/certifiedproducts/familytime_app.html.

[79] Samet Privacy, LLC. Official membership page. https://www.kidsafeseal.com/certifiedproducts/kidoz_sdk_app.html.

[80] Secureteen.com. Secureteen child app downloaded from website offers additional features. https://www.secureteen.com/ainstall/.

[81] Seleniumhq.org. Selenium automates browsers. https://www.seleniumhq.org.

[82] Sellcell.com. Kids cell phone use survey 2019 – truth about kids & phones, 2019. News article (July 15, 2019) https://www.sellcell.com/blog/kids-cell-phone-use-survey-2019/.

[83] Stiller, Nate. MAC Address Lookup. https://www.macvendorlookup.com/mac-address-lookup/.

[84] StudioTheLight. My Family Online "ailemonline" - Parental Control. https://play.google.com/store/apps/details?id=com.ailemonline.mobile.

[85] Tcpdump.org. Command-line packet analyzer. https://www.tcpdump.org/.

[86] N. Tripathi and N. Hubballi. Exploiting dhcp server-side ip address conflict detection: A dhcp starvation attack. In *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, pages 1–3. IEEE, 2015.

[87] UK Council for Child Internet Safety (UKCCIS). Child safety online: A practical guide for providers of social media and interactive services. Online article. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/487973/ukccis_guide-final__3_.pdf.

[88] Unicourt.com. Rushing et al v. The Walt Disney Company et al. Case Number: 3:17-CV-04419, filed on Mar. 8, 2017. https://unicourt.com/case/pc-db1-rushing-et-al-v-the-walt-disney-company-et-al-494632.

[89] U.S. Federal Trade Commission. COPPA Safe Harbor Program. https://www.ftc.gov/safe-harbor-program/.

[90] Verizon. Verizon 2019 Data Breach Investigation Report. https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf.

[91] Virustotal.com. Analyze suspicious files and URLs to detect types of malware. https://www.virustotal.com/gui/home/upload.

[92] W3schools.com. Html5 web storage. https://www.w3schools.com/html/html5_webstorage.asp.

[93] William Largent. Vulnerability spotlight: The circle of a bug's life, 2017. News article (Oct 31, 2017) https://blog.talosintelligence.com/2017/10/vulnerability-spotlight-circle.html.

[94] Wired.co.uk. A series of dumb security flaws left millions of EA origin users exposed. News article (June 26, 2019). https://www.wired.co.uk/article/ea-origin-account-login-security-flaw.

[95] Wireshark.org. Wireshark network analyzer. https://www.wireshark.org/.

[96] P. Wisniewski, A. K. Ghosh, H. Xu, M. B. Rosson, and J. M. Carroll. Parental control vs. teen self-regulation: Is there a middle ground for mobile online safety? In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 51–69, 2017.

[97] yandex.ru. AppMetrica tracking URL parameters. https://appmetrica.yandex.ru/docs/mobile-tracking/concepts/postback-specification.html.

[98] ZDNet.com. The latest dark web cyber-criminal trend: Selling children's personal data. News article (Mar. 27, 2019). https://www.zdnet.com/article/the-latest-dark-web-cyber-criminal-trend-selling-childrens-personal-data/.

[99] M. Zheng, P. P. Lee, and J. C. Lui. ADAM: an automatic and extensible platform to stress test android anti-virus systems. In *International conference on detection of intrusions and malware, and vulnerability assessment*, pages 82–101. Springer, 2012.

Appendix

Parental Control Tools Corpus

Table 10: List of parental control Android apps. ∗ denote versions downloaded from the company websites to enable advanced functions.

App	Installs	Package Name	Version
Circle	10K+	com.meetcircle.circle	Premium 2.8.0.2
FamilyTime	500K+	io.familytime.dashboard io.familytime.parentalcontrol io.familytime.parentalcontrol(∗) [34]	Premium 2.1.0.210 3.0.5.3196.ps 4.0.6.4209.web
FamiSafe	500K+	com.wondershare.FamiSafe	Premium 3.0.9.107
FindMyKids	10M+	org.findmykids.app	Trial 1.9.9
Kidoz	1M+	com.kidoz com.kidoz.demo.go(Fullversion*) [46]	Premium 4.0.5.8 4.0.6.3
KidsControl	1M+	ru.kidcontrol.gpstracker app.gpsme	Trial 4.0.9 k5.2.10
KidsPlace	5M+	com.kiddoware.kidsplace com.kiddoware.kidsafebrowser com.kiddoware.kidsvideoplayer com.kiddoware.kidsplace.remotecontrol com.kiddoware.kidspictureviewer	Premium 3.5.6 1.7.8 1.7.8 1.4.5 1.0.9
Life360	50M+	com.life360.Android.safetymapd	Trial 18.7.1
MMGuardian	1M+	com.mmguardian.parentapp com.mmguardian.childapp com.mmguardian.childapp(Fullversion∗) [59]	Premium 3.6.4 3.7.7 10003.9.5
MobileFence	1M+	com.mobilefence.family com.mobilefence.family.plugin(Plugin∗) [60]	Trial 2.9.3.1 1.4
Qustodio	1M+	com.qustodio.qustodioapp com.qustodio.qustodioapp(∗) [73]	Trial 180.14.2.2- family 680.14.2.2- family
ScreenTime	1M+	com.screentime.rc com.screentime	Trial 3.11.23 5.3.23
SecureTeen	1M+	com.infoweise.parentalcontrol.secureteen com.infoweise.parentalcontrol.secureteen.child com.infoweise.parentalcontrol.secureteen.child(∗) [80]	Trial 8.0.0 1.6000.5 1.7001.0

Third-Parties Analysis Results

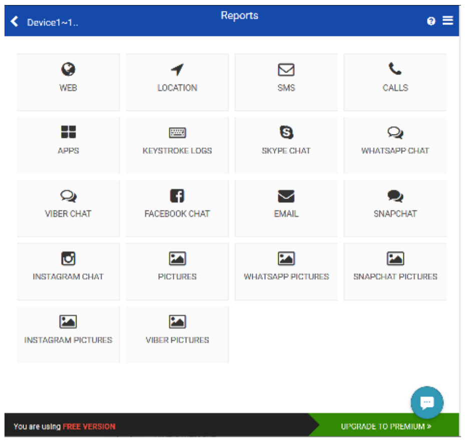
Table 12: List (count) of third-party tracking SDKs in 171 parental control Android apps.

AccountKit (6)	CleverTap (1)	Integral Ad Science (3)
AdColony (1)	Facebook Ads (10)	IronSource (4)
Adjust (8)	Facebook Analytics (38)	Kidoz (2)
AerServ (1)	Facebook Login (51)	Kissmetrics (1)
Amazon Advertisement (1)	Facebook Notifications (2)	Kochava (2)
Amplitude (9)	Facebook Places (29)	Localytics(1)
AppLovin (4)	Facebook Share (42)	Mapbox (3)
AppMetrica (7)	Flurry (10)	MaxPanel (8)
Appnext (1)	GameAnalytics (1)	Moat (6)
AppsFlyer (20)	Google Ads (47)	OneSignal (11)
Apptentive (4)	Google Analytics (62)	Splunk MINT (1)
Apteligent by VMWare (1)	Google CrashLytics (97)	Startapp (3)
BlueKai (acquired by Oracle) (1)	Google DoubleClick (44)	Tapogy (1)
Branch (9)	Google Firebase Analytics (135)	Tune (3)
Braze (formerly Appboy) (5)	Google Tag Manager (48)	Twitter MobPub (2)
Bugfender (2)	HelpShift (2)	Umeng Analytics (3)
Bugly (2)	HockeyApp (6)	Unity3dAds (1)
ChartBoost (1)	Hypertrack (4)	

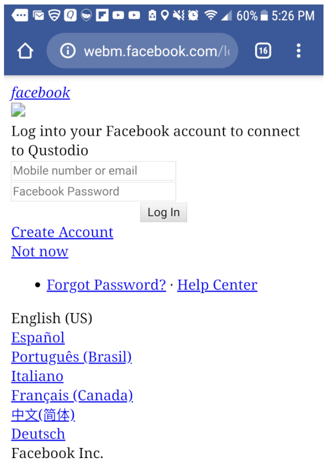
Results from VirusTotal

Table 15: 17/171 Parental control Android apps (13 available from Google Play Store) are flagged as malicious by VirusTotal. Past studies have contended that a threshold of ten AV scanners is a robust choice [17, 99]. We have reported these findings to Google Play Store in case they are not false positives. ∗: refers to app version downloaded from company website app version.

SecureTeen (Fullversion∗)	com.infoweise.parentalcontrol.secureteen.child [80]	16
SecureTeen	com.infoweise.parentalcontrol.secureteen.child	12
Kidoz	com.kidoz	6
AllTracker	city.russ.alltrackerfamily	5
Kidoz (Fullversion∗)	com.kidoz.demo.go [46]	5
mLite	com.mspy.lite	3
Boomerang	com.nationaledtech.Boomerang	2
EasyParentalControl	com.landak.gimbotparentalcontrolpro	2
Parentsaround	com.pdlp.android.app	2
Qustodio (Fullversion∗)	com.qustodio.qustodioapp[73]	2
Toddler Lock	marcone.toddlerlock	2
AppGuardian	br.com.guardian.child	1
BeCloser	com.becloser	1
ELARI	com.wherecom.elarisafefamily	1
MobileFence (Plugin∗)	com.mobilefence.family.plugin [60]	1
StepByStep	studio.wonlex.stepbystep	1
unGlue	com.unglue.parents	1



(a) SecureTeen user interface. We found an API, described in Section 5.2.1, that enables an adversary to remotely compromise any parental account and have access to the information available through this interface by knowing only the parent's email.



facebook

Log into your Facebook account to connect to Qustodio

Mobile number or email

Facebook Password

Log In

Create Account
Not now

- Forgot Password? · Help Center

English (US)
Español
Português (Brasil)
Italiano
Français (Canada)
中文(简体)
Deutsch
Facebook Inc.

(b) In Qustodio, we were able to extract the child Facebook credentials provided by the parent during the configuration of the monitoring component. The utilized Facebook login is vulnerable to SSLStript2 attack.

Figure 4: Example attack screenshots of SecureTeen and Qustodio

Table 2: Overall results for security flaws in Android apps.
✗: the app has this flaw; ✓✗ : partially compliant;
`N/A`: not applicable; `blank`: no flaw found.

App	*Insecure PII transmission*	*Share PII with third-parties*	*Insecure storage of sensitive data*	*Insecure cloud backend database*	*Lack of HSTS enforcement*	*Lack of authentication*	*Broken authentication*	*Online password bruteforce*	*Weak password policy*	*Unverified password change*	*Uninformed suspicious activities*	*Unverified parental consent*
Circle	✗	✗				✗						
FamilyTime		✗			✗	✗		✗	✗		✗	✗
FamiSafe		✗	✗	✗	✗	✗		✗			✗	✗
FindMyKids	✗	✗						`N/A`	`N/A`			
Kidoz					✗		✗	✗			✗	
KidsControl	✗	✗			✗	✗		✗	✗		✗	
KidsPlace		✗	✗		✗	✗	✗			✗	✗	✗
Life360		✗	✗		✗			✗			✗	✓✗
MMGuardian	✗	✗				✗		✗	✗	✗	✗	✗
MobileFence	✗	✗			✗			✗	✗		✗	✗
Qustodio		✗			✗		✗	✗			✗	✗
ScreenTime		✗				✗		✗			✗	✗
SecureTeen	✗	✗			✗	✗	✗	✗	✗		✗	✗

Table 3: Personal information collected by parental control Android apps through social media or registration form. `F` refers to Facebook, `G`: Google, `T`: Twitter, `R`: registration form; `blank`: no information is collected; `*`: this includes parent tweets, account settings, and accounts parent follow, mute, and block.

	Parent Information					**Child Information**					
App	***Name***	***Email***	***Phone number***	***Parent photo***	***Twitter profile info****	***Name***	***Email***	***Phone number***	***Child photo***	***Age info***	***Gender***
Circle		`R`				`R`			`R`		
FamilyTime	`RFG`	`RFG`	`R`	`FG`		`R`	`R`	`R`	`R`	`R`	`R`
FamiSafe		`R`				`RG`	`G`		`G`	`R`	
FindMyKids	`R`	`R`	`R`			`R`				`R`	`R`
Kidoz	`R`	`R`				`R`	`R`		`R`	`R`	`R`
KidsControl	`R`	`R`	`R`			`R`			`R`	`R`	
KidsPlace	`G`	`RG`		`G`							
Life360	`R`	`R`	`R`	`R`		`R`	`R`	`R`	`R`		
MMGuardian	`R`	`R`	`R`						`R`		
MobileFence	`R`	`R`	`R`			`R`			`R`	`R`	`R`
Qustodio	`R`	`R`				`RF`	`RF`		`RF`	`R`	`R`
ScreenTime	`RFG`	`RFG`		`RFG`		`R`			`R`	`R`	`R`
SecureTeen	`RFG`	`RFG`	`R`	`RFG`	`T`	`R`					`R`

Table 4: Data collected from Android child device. ✗: refers to information collected by parental apps; `blank`: no information is collected; *: Motion data was identified from the "Activity Recognition" permission in the app Manifest file [11]. This permission allows the app to detect user activities like Still, Running, Walking or Cycling. For example, Life360 uses this permission to monitor habits of young drivers. MobileFence blocks all functions on the mobile while the child is walking.

App	***Location***	***Contacts***	***Browsing history***	***Installed apps***	***App usage***	***Motion data****	***Suspicious child photos***	***Child surrounding voices***	***GPS status***	***Online status***	***Last online appearance***
Circle	✗		✗		✗						
FamilyTime	✗	✗	✗	✗	✗	✗					
FamiSafe	✗		✗	✗	✗	✗	✗				✗
FindMyKids	✗			✗	✗	✗		✗			
Kidoz			✗	✗	✗						
KidsControl	✗					✗				✗	
KidsPlace	✗		✗	✗	✗						
Life360	✗					✗				✗	✗
MMGuardian	✗	✗	✗	✗	✗		✗				
MobileFence	✗		✗	✗	✗	✗			✗	✗	✗
Qustodio	✗		✗		✗					✗	✗
ScreenTime	✗		✗	✗	✗					✗	✗
SecureTeen	✗	✗	✗	✗	✗		✗				

Table 5: Data collected from the child device related to mobile calls, SMS and social media. ❌: refers to service supported by Google Play app version; ✗: refers to a feature supported by an app distributed via the company website; ∗: the 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.

Data	***FamilyTime***	***FamiSafe***	***KidsPlace***	***MMGuardian***	***MobileFence***	***Qustodio***	***ScreenTime***	***SecureTeen***
Calls log	✗			❌	✗	✗		✗
SMS log	✗	❌		❌	✗	✗		✗
Notifications log		❌		❌	❌			✗
Whatsapp chat		❌		❌				✗
Messenger chat		❌		❌				✗
Messenger lite chat		❌						
YouTube search		❌		❌			❌	❌
YouTube playlist		❌∗	❌					
YouTube activity		❌∗	❌					
Facebook posts		❌				❌		✗
Facebook photos						❌		
Twitter		❌						
Snapchat								✗
Instagram chat		❌		❌				✗
KIK chat		❌						
Skype chat								✗
Viber chat								✗
Email				❌				✗
Keystrokes								✗
Screenshots								✗

Table 6: Sharing PII with third-parties (by Android apps)

App	**Shared PII**	**3rd-parties (number, domains [max. 3])**
Circle	Child and parent Android ID	1 (`kochava.com`)
FamilyTime	Parent email	8 (`fastspring.com`, `zopim.com`, `facebook.com`)
FamiSafe	Child name	1 (`facebook.com`)
FamiSafe	Child geolocation	1 (`googleapis.com`)
FindMyKids	Parent email	1 (`googleapis.com`)
FindMyKids	Android ID	1 (`yandex.net`)
FindMyKids	Child geolocation	1 (`openstreetmap.org`))
KidsControl	Child geolocation	1 (`openstreetmap.org`))
Life360	Child/Parent email	2 (`braze.eu`, `helpshift.com`)
KidsPlace	Child geolocation	1 (`googleapis.com`)
Life360	Child/Parent name	1 (`braze.eu`)
Life360	Child geolocation	1 (`locationiq.com`)
MMGuardian	Child browsing history	1 (`komodia.com`)
MobileFence	Child geolocation	1 (`googleapis.com`)
Qustodio	Parent name/ email	6 (`adroll.com`, `braze.eu`, `referralcandy.com`)
Qustodio	Child geolocation	1 (`googleapis.com`)
ScreenTime	Android ID	1 (`facebook.com`)
ScreenTime	Child geolocation	1 (`google.com`), and `googleapis.com`)
SecureTeen	Parent email	23 (`rlcdn.com`, `droll.com`, `ads.yahoo.com`)
SecureTeen	Child browsing history	1 (`komodia.com`)
SecureTeen	Child geolocation	1 (`google.com`)

Table 7: Network devices vulnerabilities

App	**Insecure PII transmission**	**Share PII with third-parties**	**Lack API authentication**	**Insecure storage of sensitive data**	**Faulty setup procedure**	**Developer SSH access**	**Vulnerable device**	**Vulnerable backend**	**Weak password policy**	**Uninformed suspicious activities**
Circle Home plus		✗	✗					✗		✗
KoalaSafe	✗		✗		✗		✗	✗	✗	✗
KidsWifi	✗			✗		✗		✗		
Blocksi	✗					✗	✗	✗	✗	✗
Roqos										

Table 8: Windows applications reports communication.∗: accepts both SMTP or SMTPS.

Application	Freq. of communication (per day)	**Protocol used**
Qustodio	288	HTTPS
Kaspersky	72	HTTPS
Dr. Web	288	HTTPS
Norton	144	HTTPS
Spyrix	960	HTTPS
Kidswatch	1	HTTP
KidLogger	576	HTTPS
Kurupira	N/A	SMTP*

Table 9: PII collected parental control Windows applications.
✗: refers to information collected by Windows applications;
`blank`: no information is collected; `-`: not working as intended;
`*`: premium feature.

Application Name	***Browsing history***	***Computer usage***	***Programs usage***	***Chat logs***	***Social media activities***	***Screenshots***	***Keystrokes***	***Webcam capture***
Qustodio	✗	✗	✗	✗*	✗*			
Kaspersky	✗	✗	✗		✗*			
Dr.Web	✗	✗						
Norton	✗	✗			✗*			
Spyrix	✗	✗	✗	✗		✗	✗	✗
Kidswatch	-	✗	✗	-				
KidLogger	✗*	✗	✗	✗*		✗*		
Kurupira	✗	✗	✗			✗		

Table 11: List of parental control Windows applications

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 13: List (count) of third-party domains through analyzing collected traffic from parental control Android apps installed on child device (dynamic analysis of the 13 Android apps). Third-parties domains that prohibit developers from using their services in children's apps [75]

.

2mdn.net (1)	google.ae (1)	phonedata.me (1)
3lift.com (1)	google.ca (2)	pippio.com (2)
adjust.com (2)	google.com (11)	prismic.io (1)
adnxs.com (1)	googleadservices.com (2)	pubnub.com (1)
adroll.com (1)	google-analytics.com (7)	referralcandy.com (1)
advertising.com	googleapis.com (6)	rlcdn.com (1)
aliyuncs.com (1)	googletagmanager.com (1)	rollout.io (1)
amazonaws.com (4)	googleusercontent.com (2)	rubiconproject.com (1)
appsflyer.com (1)	gstatic.com (2)	segment.com (1)
appspot.com (1)	helpshift.com (1)	segment.io (1)
awin1.com (1)	hotjar.com (1)	sendgrid.com (1)
branch.io* (1)	impactradius-event.com (1)	sentry-cdn.com (1)
braze.com (formerly Appboy) (1)	intercom.com (1)	sjv.io (1)
braze.eu (formerly Appboy) (1)	intercom.io (1)	spotxchange.com (1)
casalemedia.com (1)	kissmetrics.com (1)	superawesome.tv (1)
cloudflare.com (1)	kochava.com (1)	taboola.com (1)
cloudfront.net (2)	mixpanel.com (2)	withgoogle.com (1)
consensu.org (1)	mob.com (1)	wondershare.com (1)
cookiebot.com (1)	mobileapptracking.com (1)	yahoo.com (1)
crashlytics.com (9)	narrative.io (1)	yandex.net (1)
doubleclick.net (4)	newrelic.com (2)	yandex.ru (1)
facebook.com (5)	onesignal.com (1)	youtube.com (2)
facebook.net (1)	openstreetmap.org (1)	ytimg.com (1)
flurry.com* (1)	openx.net (1)	zenaps.com (1)
fontawesome.com (1)	outbrain.com (1)	

Table 14: Top 15 Android apps with the highest third-party tracking SDKs

App	**Package Name**	**SDKs**
BeCloser	com.becloser	22
GeoZilla	com.geozilla.family	15
FindMyKids (FSP)	com.fsp.android.g	14
Life360	com.life360.android.safetymapd	14
Hulahoop	com.hulahoop.android	13
My Kids Safety	com.family.tracker.kids.gps.locator.phone.free	11
Safe365	app.alpify	11
Zoemob	com.zoemob.gpstracking	11
JusTalkKids	com.justalk.kids.android	10
Keepers	com.keepers	10
KidSecurity	kz.sirius.kidssecurity	10
Limitly	com.bg.limitly	10
SafeFamily	com.mcafee.security.safefamily	10
FamiSafe	com.wondershare.famisafe	9
FamilyLocator	com.sygic.familywhere.android	9

Techniques Adopted by parental control tools

Table 16: Techniques used to monitor child activities including web filtering, phone calls, SMS, and social media. ✗: refers to service supported by Google Play app version; ✗: refers to a feature supported by an app distributed via the company website; ∗: Identified from app configuration files saved on the child device.

App	***Superuser request***	***Activate device admin***	***Monitor user actions***	***Retrieve window content***	***Observe typed text***	***Turn on explore by touch***	***Turn on enhanced web accessibility***	***Setup VPN connection***	***Install custom browser***	***Run MDM Agent***	***Read History Bookmarks***	***Komodia SDK***	***Make and manage phone calls***	***SMS Permission***	***Notification Access***	***Custom SMS app***	***Facebook Login [33]***	***YouTube OAuth [40]***	***Custom Video Player***	***Take Screenshot***
Circle		✗	✗	✗				✗												
FamilyTime		✗	✗	✗				✗						✗						
FamiSafe		✗	✗	✗	✗*										✗			✗		✗*
FindMyKids																				
Kidoz									✗										✗	
KidsControl													✗							
KidsPlace		✗	✗	✗					✗									✗	✗	
Life360																				
MMGuardian		✗	✗	✗	✗	✗	✗		✗		✗	✗	✗	✗	✗	✗				
MobileFence	✗	✗	✗	✗						✗	✗		✗	✗	✗					
Qustodio		✗	✗	✗							✗		✗	✗			✗			
ScreenTime		✗	✗	✗							✗									
SecureTeen		✗	✗	✗	✗			✗			✗	✗	✗	✗	✗					✗