Security and Privacy Analysis of Parental Control Solutions

Quentin Duchaussoy

A Thesis
in
The Concordia Institute
for
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Applied Science (Information Systems Security) at
Concordia University
Montréal, Québec, Canada

November 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Quentin Duchaussoy**
Entitled: **Security and Privacy Analysis of Parental Control Solutions**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Information Systems Security

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Jeremy Clark

_____ Examiner
Dr. Suryadipta Majumdar

_____ External Examiner (BCEE)
Dr. Anjan Bhowmick

_____ Thesis Supervisors
Dr. Mohammad Mannan and Dr. Amr Youssef

Approved by _____
Dr. Mohammad Mannan, Graduate Program Director

November 3, 2020 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Security and Privacy Analysis of Parental Control Solutions

Quentin Duchaussoy

For parents of young children and adolescents, the digital age has introduced many new challenges, including excessive screen time, inappropriate online content, cyber predators, and cyberbullying. To address these challenges, many parents rely on numerous parental control solutions on different platforms, including parental control network devices and software applications on mobile devices and laptops. While these parental control solutions may help digital parenting, they may also introduce serious security and privacy risks to children and parents, due to their elevated privileges and having access to a significant amount of privacy-sensitive data. In recent years, attacks and security flaws concerning this type of applications have flourished, however no systematic study for the security and privacy of these parental control solutions has been carried out to date. In this thesis, we present an experimental framework for systematically evaluating security and privacy issues in parental control software and hardware solutions. Using the developed framework, we provide the first comprehensive study of parental control solutions on multiple platforms including network devices, mobile apps, Windows applications and web extensions. We analyze a representative dataset of each type of solution and build a security and privacy state-of-the-art of each environment. Our analysis uncovers pervasive security and privacy issues that can lead to leakage of private information, and/or allow an adversary to fully control the parental control solution, and thereby may directly aid cyberbullying and cyber predators.

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Dr. Mohammad Mannan and Amr Youssef, for their support and involvement that I have been fortunate enough to benefit from throughout my master's degree. In addition to technical knowledge, they taught me the patience, rigor and abnegation that are essential to the successful completion of a research project. This thesis could not have been possible without their guidance.

I would also like to thank my fellow lab-mates from Madiba Security Research Group with whom, day by day, and through stimulating discussions, caring advice and outside activities, I have grown as an engineer and as a person. A heartfelt thanks to my French schoolmates with whom I shared the opportunity to study at Concordia University. Timothée, Antoine, William and Quentin (do not see any order of preference). We shared many memories in easy and difficult moments, and I wish that this continues henceforth.

In this regard, I would also like to express my sincere appreciation to the entire teaching staff of ESIEA Laval and more particularly to the Director of International Relations, Susan Loubet, who through her involvement made my stay at Concordia University possible and enjoyable.

I would like to thank my family who has constantly supported and accompanied me in my diverse life experiences. They always managed to convey through words the motivation and encouragement that led me to write this text today.

Finally, I also would like to acknowledge that this work was partly supported by a grant from the Office of the Privacy Commissioner of Canada (OPC) Contributions Program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Many of today's children cannot imagine their daily lives without Internet. A recent survey [64] shows that 42% of US children (4–14 years) spend over 30 hours a week on their phones; nearly 70% of parents think that such use has a positive impact on their children's development [64]. While the web could be an excellent environment for learning and socializing, there is also a plethora of online contents 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., [23, 2]); the current COVID-19 pandemic and its potential societal consequences have only increased these risks (see e.g., [69]).

To provide a safe, controlled internet experience, many parents and school administrators rely on parental control solutions that are easily accessible either for free or for a relatively cheap price. From recent surveys in the US, some forms of parental control apps/services are used by 26–39% of parents [19, 54], indicating a growing adoption of these solutions. Such solutions are also recommended by government agencies, e.g., FTC [31] in US and the Council for Child Internet Safety (UKCCIS) [68] in UK, despite

their limited effectiveness (cf. EU commissioned benchmark at sipbench.eu), and questionable morality since they, arguably, can act as surveillance tools [78]. It should be noted that this ethical and moral debate is outside the scope of this thesis.

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 [44, 55]. Aside from endangering children's safety (online and in the real-world), such leaked children's personal data may be sold by criminals (cf. [79]). Recent reports also revealed several security and privacy issues in the analyzed parental control solutions [4, 75, 27]. However, such analysis was limited to the privacy of Android apps, and only one network device, even though popular parental control solutions are used across different platforms: mobile and desktop OSes, web extensions, and network devices. Note that, unlike other vulnerable products (e.g., buggy gaming apps [76]), or non-complaint products (e.g., Android apps for children [62]), which can be removed when such concerns are known, parental control solutions are deemed *essential* by many parents and schools, and thus are not expected to be removed due to the lack of better alternatives.

1.2 Thesis Statement

Parental control solutions are products intended for underage users and process sensitive private user information to perform their operations. They must, therefore, be subjected to analysis to detect possible security and privacy problems. However, only a few academic studies have been carried out on this subject and exclusively on Android. Through this thesis, we aim to fill this gap and shed light on the security and privacy issues affecting parental control solutions in a multi-platform approach. We also investigate, as a second objective, how parental control solutions carry out monitoring operations and identify the similarities and discrepancies across the different platforms, especially in terms of range and capabilities.

In this thesis, we undertake the first comprehensive study to analyze different types of parental control hardware and software solutions. We design a set of security and privacy tests, and systematically analyze popular representative parental control solutions available in network devices, Windows and Android OSes, and Chrome extensions. While developing our comprehensive analysis framework for solutions in multiple platforms, we faced several challenges. Most parental control solutions implement various techniques that hinder traffic analysis (e.g., custom protocols). The use of proprietary firmware and code obfuscation techniques also poses challenges for static analysis. Furthermore, these solutions are expected to be used continuously over prolonged periods. Automatic, episodic and cursory security testings are likely to miss critical security flaws. Studying the long-term behavior of this type of products by actively using them for days presents obvious temporal constraints which hinders this type of analysis.

1.3 Contributions

Our contributions can be summarized as follows.

- We developed an experimental framework for systematically evaluating security and privacy issues in parental control software and hardware solutions. We especially emphasized three elements: network traffic analysis, client product static analysis and online user interface analysis. The study of network flows brought to light access control issues, insecure transmissions of sensitive data as well as the communication of private data to third parties. The static analysis complemented the dynamic analysis and highlighted flaws in the client software endangering the user. Finally, the analysis of the client interface revealed weaknesses that could lead to the client's account being compromised.

- We utilized this framework to conduct the first comprehensive study of parental control solutions on multiple platforms, including 8 network devices, 8 Windows applications, 10 Chrome extensions and 28 Android apps.[1]

- Our analysis identified 170 vulnerabilities among the solutions tested and revealed that the majority of solutions broadly fail to adequately preserve the security and privacy of their users, both children and parents.

Related Publication. The analysis discussed in this thesis has been peer-reviewed and accepted in the Annual Computer Security Applications Conference (ACSAC 2020) as a paper entitled:

Betrayed by the Guardian: Security and Privacy Risks of Parental Control Solutions. Suzan Ali, Mounir Elgharabawy, Quentin Duchaussoy, Mohammad Mannan, and Amr Youssef. 2020. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA*.

1.4 Ethical Considerations

Our study revealed many security and privacy vulnerabilities in parental control solutions. In the absence of defined guidelines for vulnerability analysis, we followed the best practices set in previous work [41, 3, 24]. To ensure the ethical aspect of our study, our analysis was performed exclusively on our own testing devices and dedicated accounts. Throughout this work, we never accessed other users' information. In addition, our assessment of the security of the backend and online interface was limited to ensure that the extra load did not affect availability. Whenever possible, we prioritized side-channel measurements to assess backend vulnerabilities.

[1]The Android analysis are separated into 2 sets. An initial analysis of 13 Android apps was performed by a co-researcher. Thus, it will only be briefly mentioned in the following parts.

As part of responsible disclosure, we contacted the developers of the solutions we analyzed and shared our findings, including proof-of-concept scenarios and possible fixes. Four months after disclosure, only ten companies responded (one Windows application, two network devices and seven Android apps). We received seven custom and three automatic replies. Notable changes after the disclosure: one Android app deprecated their custom browser; Another fixed the Firebase database security issue; and a third enabled HSTS on their server.

1.5 Outline

This thesis is organized as follows: chapter 2 introduces the background information required to understand the technical context of the analysis, the decisions made, the challenges faced and the conception of the methodology. Subsequently, we describe and discuss prominent related works relative to the analysis of parental control solutions, and security and privacy studies conducted on the various platforms analyzed. Chapter 3 presents the framework developed and used to analyze a dataset of parental control solutions covering Windows application, network device, Android apps and Browser extension. We also describe the threat model observed and the security and privacy issues considered. Additionally, we provide insight on the selection of the solutions analyzed. Chapter 4 groups the results obtained based on the aforementioned methodology. We divided this part by platform and related threats as stated in Chapter 3. Chapter 5 offers a conclusion to the thesis and proposes recommendations and future research perspectives on this topic.

Chapter 2

Background and Related Work

In this chapter, we present the background associated with this thesis and the related work literature. We define the terminologies used in this thesis and a type of attack performed in the context of our analysis. We cover the different platforms studied and the monitoring techniques featured by parental control solutions. We also discuss the related work regarding security and privacy analysis previously conducted on each platform.

2.1 Background

2.1.1 Terminology

In what follows, we use the term *parental control solutions* to cover different types of parental solutions: network devices, Chrome extensions, Windows applications and Android apps.

Similarly, *personally identifiable information (PII)* refers to any information related to the user as defined by the US FTC and Office of the Privacy Commissioner of Canada. It may concern the user identity (name, email, address, unique identifiers) or behavior (web activity: URL visited, computer usage).

We consider as *third party* any entity that is not directly related to a parental control solution; this includes but is not limited to trackers and advertisers.

2.1.2 SSLStrip Attacks

SSLStrip attacks refer to the use of techniques to strip away the security provided by HTTPS, exposing user private information in plaintext. We tested the analyzed parental control solutions against this type of attack in our study. This type of attack was first presented in 2009 by Moxie Marlinspike at the Black hat conference [45]. In essence, the original attack consists in exploiting the transition from HTTP pages to HTTPS. The HTTP content being unencrypted, an attacker can modify HTTPS hyperlinks present in the HTTP page and trick the user in sending private information. This version of the attack was especially prevalent in the context where most of the website only provided HTTPS on their login pages.

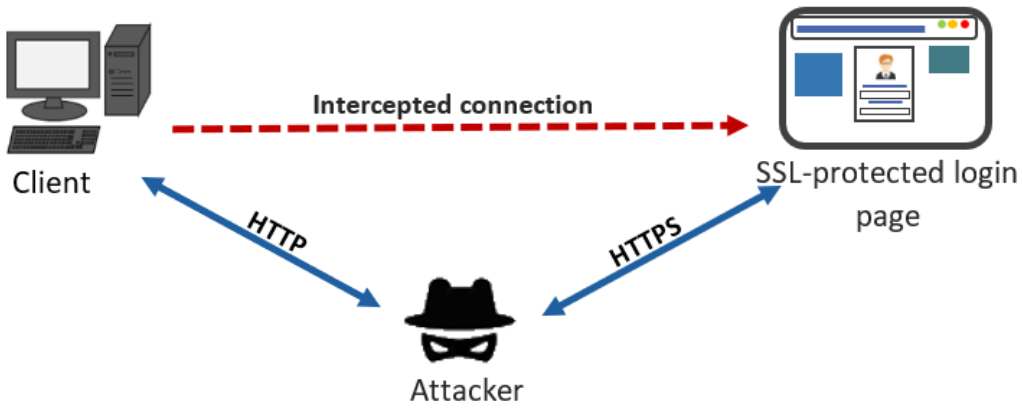


Figure 1: SSLStrip attack.

In consequence, the IETF developed in 2012 the HTTP Strict Transport Security (HSTS), a security mechanism designed to circumvent the SSLStrip class of attacks. This mechanism allows web servers to communicate to the web browser that any subsequent connection must be issued over HTTPS. The web server response header contains the `Strict-`

`Transport-Security` field which specifies the duration of the security constraint.

In 2014, Leonardo Nve Egea proposed an enhanced version of the attack at the Black Hat conference. Named SSLStrip+ or SSLStrip2, this newer version was able to bypass the HSTS security mechanism. In practice, the attacker exploits a flaw in the HSTS setting. In the absence of the `includeSubDomains` flag, `www.example.com` is protected by HSTS but `wwww.example.com`, which is not recognized, is not. In SSLStrip+, the attacker bypass HSTS this way, when the client attempts to resolve `wwww.example.com`, the attacker uses a DNS resolved to mirror `www.example.com`.

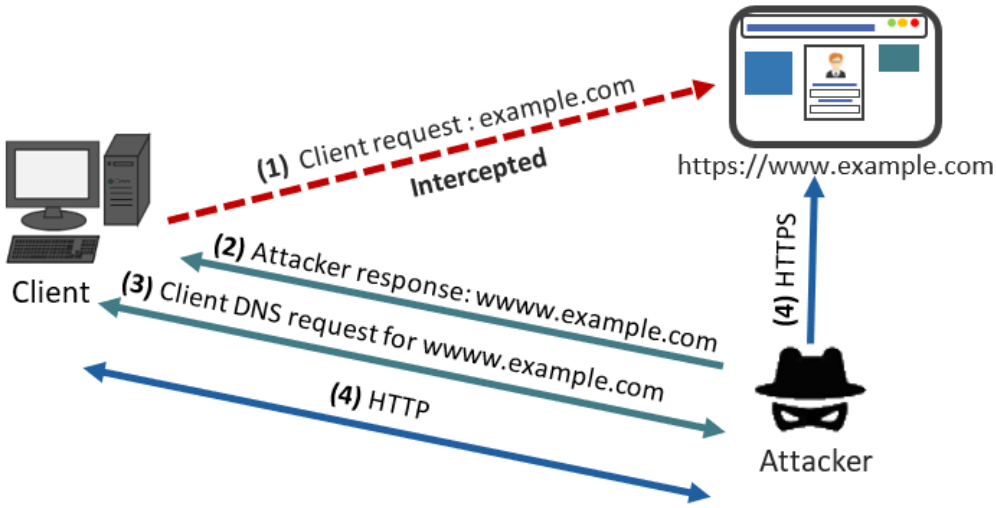


Figure 2: SSLStrip+ attack.

As explained above, the `includeSubDomains` flag should effectively defend the user against this enhanced version. However, our analysis showed that most parental control solutions do not include HSTS or it is not properly configured.

2.1.3 Main Types of Parental Control Solutions

Parental control solutions across and within the different platforms can be categorized in several ways. Nevertheless, we can distinguish two main families of such solutions: host-based solutions and network-based solutions. The difference between these two categories lies essentially in their capabilities. The host-based solutions are directly installed on the monitored device, whereas the network-based solutions monitor the devices remotely. This setup difference affects the data available and the range of operation the solution can perform.

Host-based parental control solution. This is the most common family of parental controls. Intrinsically linked with the democratization of home computing, parent control solutions began to appear in the early 1990s. Initially limited to personal computers, the market has now opened up to mobile and tablet devices, as well as web browsers (via extensions). In practice, host-based solutions have access to a wide variety of information and control tools, depending on the restrictions of the platform and their permissions. In particular, they can collect and use information about the usage of the host platform (e.g., programs launched, resource utilization), user inputs (e.g., keystrokes, search engine queries) and media access (e.g., enable webcam, record user's screen activity). In addition to the data collection capabilities, they may also enforce parenting through various operations (e.g., PC or device lock, data interception).

Network-based parental control solution. In contrast, network-based parental control solutions have a much more limited access to user information. Their only available sources of information is the network flows to and from the user device. Therefore, they are unable to control and monitor launched applications and requested local resources. We can further differentiate the network-based solution in two types: network devices, operating on the local network, and DNS-based solutions, which are remote DNS servers incorporating parental control features. Located on the path between the user device and the online

resources, the network-based solutions rely primarily on DNS (protocol used to convert human-readable information - URL, in machine readable one - IP address) queries to identify the requested content and determine its suitability. In its most commonly used version (DNS-Over-HTTPS and DNS-Over-TLS excluded), this protocol does not protect the confidentiality of the requested resource, the information is sent unencrypted. Thus, parental control solutions can read the packet, identify the requested resource and determine if the content is suitable. Network-based solutions naturally feature a much restricted set of operations, usually limited to the modification and drop of the out-going DNS requests, or the in-going requested resources.

In this thesis, we conduct an analysis on four different platforms: Windows applications, Android apps, web extensions and network devices. Because of the inherent limitations of our study in terms of scope of observation we are restricted in our operations and observations to the client side of the solutions, we cannot directly analyze how private data is handled once received by the server. Hence, we excluded DNS-based solutions, which feature a fully remote parental control engine, from our analysis. We also decided not to analyze OS-based parental control features (e.g., Windows 10 parental control), their operation is too intrinsically tied to the functioning of the operating system itself, thereby hindering both network flow analysis and reverse engineering.

In what follows, we briefly discuss some common techniques used by parental control solutions to perform parental monitoring.

2.1.4 Monitoring Techniques

Parental control solutions generally allow the parent to remotely control the child device, perform web filtering, and monitor online activities. However, the capabilities of these solutions vary greatly from one platform to another and within the same platform. Thus, it is important to get familiar with the different techniques used. First of all, not all parental

control software provides the same features. The monitoring means are highly related to the usage of the supervised device. Likewise, the permissions granted and monitoring scope differ from one platform to another. For instance, Android parental control apps almost all support geolocation tracking, whereas this feature is not available for Windows applications or network devices. Similarly, parental control browser extensions focus exclusively on monitoring online accessed resources, as they do not have the ability to monitor local resources.

The following list of monitoring techniques is derived from the product documentations and our observations from installation procedure and analysis of these solutions. As the techniques vary significantly across platforms, we chose to regrouped here as such.

Network devices. The parental control network devices only have access to the local network to enforce digital parenting. No one of the network devices analyzed in our study required to install a certificate in the child device trusted root certificate store. They could only monitor plaintext data and were unable to inspect the content of encrypted traffic.

The devices 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, also known as ARP poisoning is a MITM technique that enables the network device to impersonate the internet router on the local network. The device achieves that by sending forged ARP packets that bind the router's IP with the network device's MAC address. As a result, all the local network traffic is routed through the device before going to the internet router.

  In here, we explain an example of ARP poisoning. The parental control device sends a specially crafted ARP packet to the child device. With the type "is-at", the parental control device informs the receiving device that it owned the "ip_source" of the gateway (binding the IP address and the Mac address). Following it, the child device will

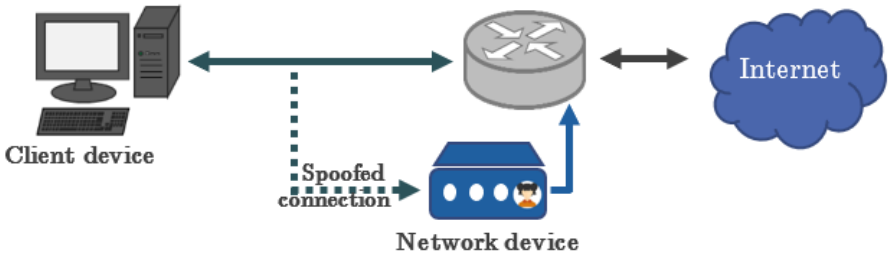


Figure 3: ARP poisoning method used by network devices.

send messages means for the gateway to the parental control device.

```
Local network :
192.168.1.1 : gateway
191.168.1.10 : Targeted device
192.168.1.25 : Parental control device
```

```
ARP packet :
ip_source= 192.168.1.1
mac_source= [Parental control MAC_address]
ip_destination= 192.168.1.10
type = is-at
```

- Alternatively, the network device may create an explicit access point exclusively for children to enforce parental control filtering on all devices connected to it.

Android apps. Android apps rely on several Android-specific mechanisms. To provide insights about the techniques used, we extracted the permissions used by the Android solutions analyzed in our study and present some of them in Table 1.

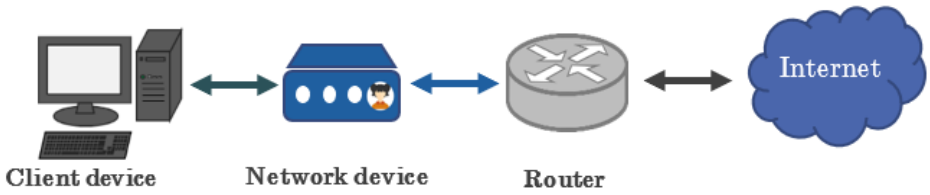


Figure 4: Access point method used by network devices.

Technique	*Becloser*	*Geozilla*	*Iwawa*	*Kids home*	*My kids Safety*	*Tittle parental control*	*Fun control*	*Sentry*	*Safe Lagoon*	*Locategy*	*Easy parental control*	*Keepers Child Safety*	*Safezone*	*Bosco*	*Saferkid*
Activate device admin						●		●	●	●	●	●		●	●
View and control screen						●		●	●	●	●	●		●	●
View and perform actions						●		●	●	●	●	●		●	●
Apps usage statistics			●	●		●	●	●	●	●		●			
Custom browser			●						●						
Custom video player			●												
Manage phone calls						●		●	●	●				●	●
SMS permission									●						
Notification access									●					●	●
Access location	●	●			●	●		●	●	●	●	●	●	●	
Facebook login					●								●	●	
Instagram login														●	
Screen capture									●						
Access storage			●			●		●	●					●	
Video/Audio recording						●								●	
Display over other apps			●				●		●			●			
Modify system setting											●				
Change home launcher				●											

Table 1: Techniques used by Android apps to monitor child activities.

- *Device administration* provides several administrative features at the system level. This permission allows the apps, among others, to lock the device, factory reset, install certificates and manage the storage encryption. The main usage of this permission by parental control apps is to prevent the child to uninstall the application.

- *Android accessibility service* allows apps to perform various operations. It includes subsidiary permissions such as `view and control screen` and `view and perform actions`. Enabling accessibility services gives the apps the possibility to monitor user actions (via received notifications), user interactions with an app, capturing and retrieving window content, logging keystrokes. It even allows the monitoring app to control visited online resources by injecting JavaScript code into the requested web pages. The range of operation available explains its prevalence among the analyzed apps as it provided most of the data and control means necessary for the apps to operate.

- *Apps usage statistics* permission gives access to the device usage and history. It allows parental control apps to monitor the usage of the other apps used on the device.

- *Custom browser and video player* doesn't require specific permission. They are used to closely monitor the resources accessed by the child.

- *Manage phone calls* is a self-explanatory user permission. It allows an app to query the current cellular information and status of on-going calls.

- *SMS permission* authorizes the granted app to read SMS sent and received.

- *Notification access* enables Android apps to read or dismiss all notifications displayed in the status bar; these notifications widely used by messaging apps, among others, may include personal information such as contact names and communication content.

- *Access location* permission (access_fine_location) gives the app full control to get the device location; It permits to use GPS and WiFi and mobile cell data to precisely locate the user. This permission is the most requested among our tested apps. The large presence of geo-tracking apps among mobile parental control apps can be explained by the need for parents to constantly being aware of the child's location.

- *Social media login* is another technique used by parental control apps. Leveraging on specifically developed social media API, the app gets access to the child's social activities.

- *Screen capture* permission (screen casting) allows apps to capture the user screen and all the information displayed on, and send it elsewhere. This is a specific permission which in Android 10 need to be grant every time it is being used.

- *Access storage* is a generic permission, it is necessary to access the phone gallery and saved media (picture, video and music).

- *Audio recording* allows the app to get data from the embedded phone microphone. This can be used to retrieve the ambient surrounding sound.

- *Display over other apps* is a permission used to enforce parenting by Android apps. It permits to draw an overlay over any activity windows (regular screen, system windows are still displayed in foreground) and thus to deny access to specific apps or resources in complement with identifying permission.

We have also encountered one solution that requires to change the *home launcher* to its own. The home launcher consists in screens, shortcuts and widgets. Effectively this technique was used to trap the user within the parental control app which played the role of a safe environment app.

Windows applications. As opposed to Android parental control apps, Windows applications do not require specific permission and operate with more inherent privileges. From our observation they use the following techniques:

- *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.
- *Application monitoring*: user applications are monitored for their usage and duration.
- *User activity monitoring*: some Windows applications take screenshots, record keystrokes, and access the webcam.

Chrome extensions. The web extensions operate within the boundaries allowed by the browser. They are only able to monitor online resources displayed via the web browser. Chrome web extension rights are regulated on a very similar fashion to Android. Extensions require permissions granted by the user at installation or run-time to operate. We have extracted from the analyzed parental control extensions the requested permissions and their associated description and present the main ones in Table 2.

Permission	*Blocksi Web Filter*	*Parental control*	*TinyFilter*	*Porn Blocker*	*Adult Blocker*	*Anti-porn addon*	*MateCode Blocker*	*MetaCert*	*FamilyFriendly*	*Kids Safe Web*
Tabs	●	●	●		●	●	●	●	●	●
WebRequest	●	●	●		●	●	●			
RequestBlocking	●	●	●		●	●	●			
WebNavigation	●	●					●			●
Storage	●	●		●	●	●	●		●	●
ContextMenus		●				●				
nativeMessaging		●								
Cookies		●				●	●	●		
Background								●		●
Management										●
http://*/*	●	●	●	●	●					
https://*/*	●	●	●	●	●					
<all_urls>	●						●	●	●	

Table 2: Permissions requested by Chrome web extensions to monitor child online activities.

- *Tabs* is a powerful permission that gives read and write rights on the Tab object fields. This object is used by the browser to describe a user tab. Among other elements, its fields contain the Tab `ID`, the `openerTabID` (ID of the tab that opened it), the `URL` (URL of the main frame), the `pendingURL` (URL requested before it has committed) and the Tab `Title` (associated name of the main frame URL). This information effectively provides the monitoring extensions way to detect unsuitable

content and enforce parenting.

- *WebRequest* and *RequestBlocking* are required to use the chrome.webRequest API. This API can be used to observe, analyze, intercept and modify request in-flight. RequestBlocking must be granted to use the above mentioned API in a blocking way.

- *WebNavigation* permission gives access to the chrome.webNavigation API used to get further information about the status of web requests in-flight. On Chrome, the navigation requests go through four successive steps: `onBeforeNavigate`, `onCommitted`, `onDOMContentLoaded` and `onCompleted`. In combination with the WebRequest permission, this permission can allow fine monitoring.

- *Storage* permission allows to use the chrome.storage API to write, read and track modification in user data.

- *ContextMenus* gives rights to use the eponymous API to modify Chrome's context menu (e.g., frame, media, hyperlink and pages).

- With *NativeMessaging*, the extensions can communicate with native application registered as native messaging host.

- *Cookies*'s API authorize the extension to query and modify user cookies.

- *Background* is an interesting permission. It makes the Chrome run as soon as the user logs to the computer and continue even after Chrome windows are closed. The extension with the associate permissions are naturally running along with Chrome.

- *Management* gives permission to the extension to monitor the other installed extensions.

- *Extension Scope:* Chrome extensions consist of two different categories of scripts: *Background script*, running in background and used to perform long term operations and managing the extension general functioning, and *Content script* invoked on page matching a certain pattern. The "match patterns" consist in an URL starting with a specific scheme (HTTP(s), FTP, etc.) and possibly wildcard characters. *http://*/** and *https://*/** are two matching patterns that determine the scope of the extension. For instance, *http://*/** only allows the extension to run content script on http pages. <*all_url*> is a pattern that match any URL.

2.2 Related Work

Over the past years, several parental control solutions have made the news for security and privacy breaches. For instance, the teen-monitoring app TeenSafe leaked thousands of children's Apple IDs, email addresses and passwords [44]. Family Orbit exposed nearly 281 GB of children data from an unsecured cloud server [55]. In 2019, a privacy flaw in Kaspersky anti-virus and parental control application was found [25]. This application included a script to perform content checking on each page intercepted by a TLS proxy. However, some unique IDs were also included in the process, allowing the website to track the user. In 2010, EchoMetrix settled US FTC charges for collecting and selling children's information to third parties through their parental control software [26].

Parental control analysis. From an academic point of view, studies on parental control solutions have focused on the fast growing field of Android apps with very few work done on other platforms.

Between 2015 and 2017, researchers from the Citizen Lab (citizenlab.ca), Cure53 (cure 53.de), and OpenNet Korea (opennetkorea.org) published a series of technical audits of three popular Korean parenting apps mandated by the Korean government, Smart Sheriff [4, 5] , Cyber Security Zone [6] and Smart Dream [7]. 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. Similarly, Smart Dream allowed unauthorized access to children's messages and search history.

Feal et al. [27] recently studied 46 parental control Android apps for data collection and data sharing practices, and the completeness and correctness of their privacy policies. Their approach focused more on personal information leakage and privacy policy analysis. They relied on the Lumen Android app (see `https://haystack.mobi/`) for their analysis, which is unable to analyze target apps with VPN or certificate pinning. Parent apps and dashboards were also excluded from their analysis. In contrast, our analysis framework doesn't present such limitations and evenly focus on security and privacy related issues as the former negatively affect the latter. Consequently, we are able to identify new critical security issues (e.g., leakage of plaintext authentication information), even among the apps analyzed by Feal et al.

Wisniewski et al. [78] conducted an analysis on 75 parental control apps to determine the dominant feature and parenting strategies among them. They evaluated 42 features and shown that most apps value control over self-regulation strategies. In addition, they found that apps boast the use of privacy invasive techniques.

Marsh [46] carried out an examination of the perception of privacy online risk and digital privacy. Through interviews with a panel of parents and children and using prepared scenarios they attempted to measure the effectiveness and usability of parental control apps. Additionally, they identified a list of seven parenting strategies mean to equally satisfy the need for online safety and the right of digital privacy.

Reyes et al. [62] analyzed children Android apps for COPPA compliance. Out of 5855 apps, the majority of the analyzed apps were found to potentially violate COPPA, and 19% were found to send PII in their network traces. Their large scale analysis built a ground truth of the COPPA compliance of Android apps. Several apps analyzed are parental control

apps.

Web extensions privacy and security analysis. Over the past decade, Chrome extensions have made news for security and privacy issues [74, 33, 30]. The security analysis of web extensions is an on-going topic and was the subject of several studies.

In 2017, Starov et al. [66] conducted a large scale study on privacy leakage caused web extensions. They relied on simulated user interaction and keyword search and found that several hundred extensions leaked private information. More interestingly, they found that the majority of the extension leaked information "accidentally" via the Referer header by injecting third-party content.

In 2017, Weissbacher et al. [73] designed an automated solution to identify privacy-violating extension relying solely on the analysis of network flows. The underlying idea is that if the user history increases the leaking extension data sharing will proportional increase. Through their tool, they managed to identify a few rogue extensions, including Web of Trust, which were immunized against formed detection techniques.

In 2018, Chen et al. [14] approached the problem of privacy leakage differently. They performed an enhanced taint analysis by customizing the browser JavaScript engine to identify leaking private information extensions. We used in our analysis the tool they made available, Mystique (`https://mystique.csc.ncsu.edu/`), to compare and cross-check our results.

In 2019, A. Beggs et al. [10] looked at wild extensions. These web extensions are not referenced on the Chrome Web Store itself but downloadable from external sources. The researchers demonstrated that these unregistered extensions posed serious security problems. They focus on the various methods used by these wild extensions to hide and trick users into installation (e.g. social engineering). Subsequently, they relied on Mystique to detect privacy issues.

Our analysis differs from the previous ones in that we have been interested in a very

specific type of extension: parental control extension. We especially focus on the comprehensibility of the study. We relied on previous works to determine the elements likely to lead to privacy leakage.

Windows applications privacy and security analysis. To our knowledge, Windows parental control applications have been only studied with respect to the security of their TLS proxies.

In 2016, De Carné de Carnavalet et al. [22] analyzed a dozen windows applications (including a few parental control applications) performing some kind of TLS interceptions. They discovered that many products put users at risk by degrading or ignoring certain security tests that are essential to the TLS ecosystem. We drew inspiration from their analysis to study our TLS proxy applications and we enhanced it by considering further security and privacy vulnerabilities.

Network device privacy and security analysis. Similar to Windows applications, very few studies have been carried out on parental control network devices and this field of research remained mainly unexplored.

In 2017, Cisco Talos analyzed the Disney Circle parental control network device and found 23 different security vulnerabilities [75]. We also analyzed Circle, among other devices, but conducted our analysis on a newer version released in 2019. This version is based on the insights from Cisco Talos's analysis.

Interestingly, several studies have been conducted on smart toys. This type of IoT device, equipped with network capability, shares some commonalities with parental control devices. In particular regarding the intended user and the legal constraints resulting from it, as well as the types of data that may be leaked. In 2018, Shasha et al. [65] proposed a taxonomy of the flaws specific to its type of devices and a methodology to detect and analyze them. They showed that behind the harmless toy appearance, such devices presented critical security and privacy flaws, threatening the digital and physical security of

their owner.

Conclusion. In contrast to previous work, we conduct a comprehensive, systematic study of security and privacy threats in parental control solutions across multiple platforms: desktop (Windows), web browser (Chrome extensions), mobile (Android) and stand-alone network devices, as popular solutions are available in all these platforms. Our analysis shed light on a broader picture of security and privacy risks of parental control solutions. Compared to existing Android app studies of parental control apps, our framework was also more in-depth and inclusive.

Chapter 3

Framework for Analyzing Security and Privacy Issues in Parental Control Solutions

In this chapter, we specify the security and privacy issues we focused our analysis on and define the threat model considered in this thesis. Then, we present the criteria used to select the solutions analyzed and describe the experimental framework developed to systematically evaluate parental control solutions.

3.1 Potential Security and Privacy Issues

We define the following list of potential security and privacy issues to evaluate parental control solutions. All the tests were performed using only our own accounts when applicable. This list is initially inspired by previous work [4, 60, 22, 65], and then iteratively refined by us.

1. *Vulnerable client product*: A parental control product and features (including its update mechanism) being vulnerable. This flaw may allow sensitive information disclosure (e.g., via on-device side-channels), or even full product compromise (e.g., via arbitrary code execution).

2. *Vulnerable backend*: The use of remotely exploitable outdated server software, and misconfigured or unauthenticated backend API endpoints. This vulnerability can lead, among other issues, to sensitive information leakage and the bypass of secure access control mechanisms.

3. *Improper access control*: Failure to properly check whether the requester owns the account before accepting queries at the server-end. A textbook case of this type of issue is the insecure direct object reference [52].

4. *Insecure authentication secrets*: Plaintext storage or transmission of authentication secrets (e.g., passwords and session IDs).

5. *SSLStrip attack*: The parental control solution's online management interface is vulnerable to SSLStrip attack, possibly due to lack of HSTS enforcement (cf. [43, 42]).

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

7. *Online password brute-force*: No defense against unlimited login attempts on the online parental login interface.

8. *Uninformed suspicious activities*: No notifications to parents about potentially dangerous activities (e.g., the use of parental accounts on a new device, or password changes).

9. *Insecure PII transmission*: PII from the client-end is sent without encryption, allowing an adversary to eavesdrop for PII.

10. *Unjustified PII exposure* : PII collection and sharing (from client devices) with third parties or with the parental control solution when it appears to be unnecessary to perform the monitoring operation.

3.2 Threat Model

We consider the following attacker types with varying capabilities.

- *Local network attacker:* an attacker with direct or remote access to the child device local network. This attacker can eavesdrop communication and directly interact with the child device. She can consist in a malicious (or compromised) application installed on the child device or a physical person connected to the child local network.

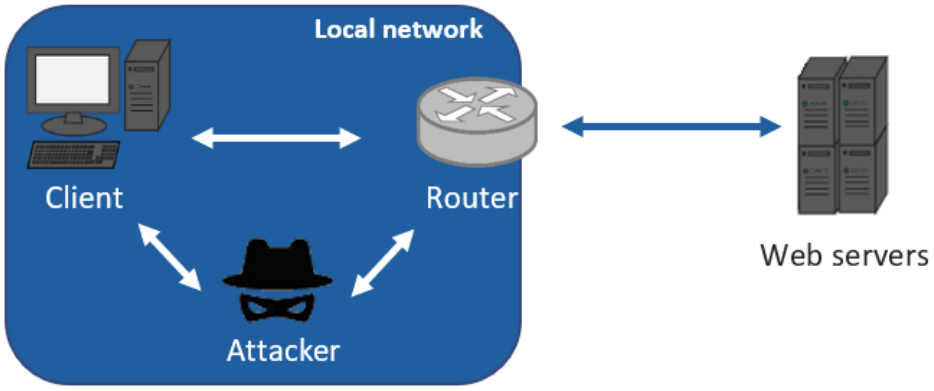


Figure 5: Local network attacker.

- *On-path attacker:* a man-in-the-middle attacker between the home network and either the solution's backend server. This attacker can eavesdrop, replay, modify, and drop communication. She can consist in a compromised router or a malicious or compromised ISP.

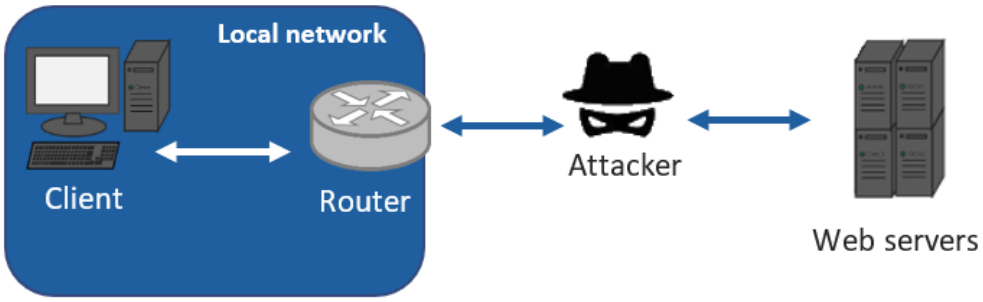


Figure 6: On-path attacker.

- *Remote attacker:* any attacker who can connect to a solution's backend server. This attacker can interact with the solution backend via API calls and online user interface. She doesn't have the capability to directly collect user information but she can exploit publicly available sources (e.g., marketing database). Additionally, she performs realistic brute force attacks on weak passwords and enumerates short and guessable solution ID.

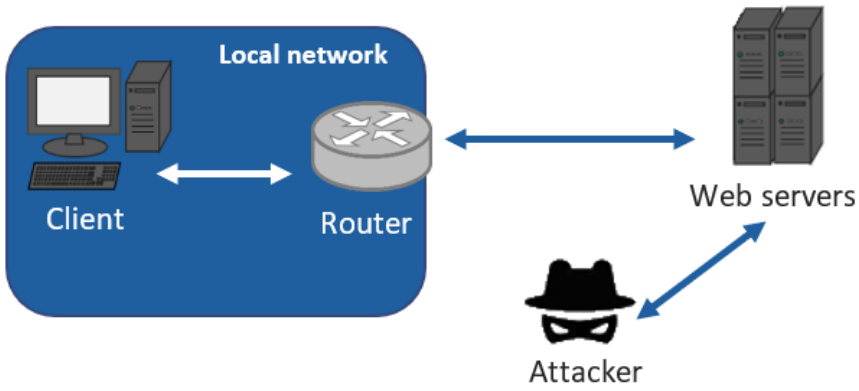


Figure 7: Remote attacker.

We exclude from our threat model any attacks requiring physical access to either the child/parent device. An attacker with physical access to the user devices has a variety

of avenue to bypass security checks and it renders useless even the most advanced ones. Consequently, we chose to not consider them.

3.3 Selection of the Solutions

We chose solutions used in the most popular computing platforms for mobile devices (Android), personal computers (Windows), web browser (Chrome), and selected network products from popular online marketplaces (Amazon). As of October 2020, current market shares according to one estimate (`https://gs.statcounter.com`) are Android 72.9%, Windows 76.3% and Chrome 63.8%.

3.3.1 Windows Applications

In the absence of a reliable method to measure popularity or obtain the number of downloads of a Windows application, we relied on rankings and reviews provided by specialized media outlets (e.g., [13, 53, 37]). The Windows applications chosen are: Qustodio, Kaspersky, Dr. Web, Norton, Syprix, Kidswatch, KidLogger and Kurupira. Due to the initial inspiration of the project [22] and to the criticality it brings, we have prioritized the analysis of software that operates a form of content checking via a TLS-interception proxy. In the absence of a more accurate system to measure the popularity and in addition to the specified reviews, we used Alexa services to collect information on website usage associated with the selected applications, see Table 3. However, this information should be treated with caution as some sites are not solely dedicated to the parental control application and some solutions have an online interface while others do not.

Application	# of visits/day	Main countries	World Alexa rank
Qustodio	27k	US	85,673
Kaspersky	1.4M	IN, US, RU	2,114
Dr. Web	84K	US	40,515
Norton	6.4M	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 3: List of parental control Windows applications.

3.3.2 Network Devices

Concerning the network devices, we selected and bought eight devices from Amazon. Amazon presents different categories of products (e.g., electronics, network router) and each category has its own internal ranking. However, parental control is not a category and we found parental control solutions across different categories which prevent us from using it to select the most popular ones. Another possibility would have been to rely on customer reviews and ratings, however, the shortcomings of this technique are very noticeable as reviews are easily manipulable. In consequence, we opted to rely on the Amazon store SEO. This decision is primarily motivated by a desire to mimic a normal user selection of parental control products. We searched for “Parental Control" on the Canadian and US Amazon store and selected relevant products among the top searches. Our criteria included the presence of parental control as a primary feature and a reasonable price (<350 C$). We analyzed eight network devices: Circle home plus, Blocksi, KoalaSafe, KidsWifi, Roqos core, Bitdefender box , HomeHalo and Fingbox, see Table 4.

Device	Version
Circle Home Plus	3.10.0.2
KoalaSafe	1.26825
KidsWifi	N/A
Blocksi	N/A
Bitdefender	2.1.66.4
Roqos	N/A
HomeHalo	1.0.0.8
Fingbox	0.5-2ubuntu4

Table 4: List of parental control devices and their firmware versions.

3.3.3 Browser Extensions

We decided to focus on Chrome web extensions. With 62.5% of the market share, Chrome is currently the most representative web browser. We chose all the extensions in the Chrome Web Store and ignored Wild extensions (extension downloaded from third parties). The reasons behind this decision are that wild extension are not indexed and present higher security risk compared to extensions approved by Chrome. Moreover, from a regular user perspective, the selection of extension from the Chrome Web Store is more logical. We faced similar issues described in Sec. 3.3.2 to select and measure the popularity of parental control extension (e.g., lack of categories and unreliableness of user review and notation). The Chrome Web Store displays the total number of downloads for each application in the form of a threshold (e.g., +100 + 1000, +2000 downloads). A higher number is not a direct evidence of a higher number of active users, but it still provides valuable insight on its usage. We chose ten extensions among the most “popular" (more than 1000 downloads) and relevant results sorted by the store’s search engine for the term “parental control". Most of the names of the chosen extensions are some sort of combination of parental control, adult and blocker. Therefore, in order to distinguish the extensions more easily, we have associated a part of the extension name with the developer one. The extensions analyzed are listed in Table 5.

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

Table 5: List of parental control Chrome extensions.

3.3.4 Android Apps

On the mobile apps side, we focused on Android, the most popular operating system with 72.6% market share. We obtained a total of 462 applications using the following search terms: "parental control", "family tracker". From this initial dataset, we selected 158 applications with 10k downloads or more. These applications were then automatically analyzed. Excluding the unresponsive and irrelevant ones, we came up with a total of 153 applications analyzed; 33% of applications intended for children (child apps), 15% to parents (parent apps), 52% of applications that needed to be installed on both the child's and the parent's device (shared apps). Subsequently, we performed an in-depth analysis on two sets of a dozen solutions. The first series carried out by a student of the laboratory included 13 solutions, the second series that I analyzed included 15 solutions. Each solution could include a child, parent application and an online user interface. The analyzed solutions are listed in Table 6.

Solution	Installs	App package name	Version
Becloser	10M+	com.becloser	3.0.19
Geozilla	5M+	com.geozilla.family	6.10.8
Iwawa	1M+	com.sencatech.iwawa.iwawahome	5.5.6
Kids home	50K+	com.arolle.kidshome	3.0
My kids Safety	10K+	com.family.tracker.kids.gps.locator.phone.free	1.5.0
Tittle parental control	10K+	com.bluecube.parentalkit	4.9.7
Fun control	10K+	me.funcontrol.app	1.0.191003
Sentry	100K+	com.sentry kid / parental	2.6.4
Safe Lagoon	100K+	com.safelagoon.parenting	4.2.151-b
Locategy	100K+	com.locategy.family	1.97.5
Easy parental control	10K+	com.landak.gimbotparentalcontrol	1.2.4
Keepers Child Safety	50K+	com.keepers	1.1.60
Safezone	500K+	com.omrup.cell.tracker	1.63
Bosco	100K+	com.bosco.boscoApp	75.40
Saferkid	50K+	com.saferkid.monitor	1.0.41

Table 6: List of parental control Android app (Set 2).

3.4 Methodology

We combined dynamic (primarily traffic and usage) and static (primarily code review/reverse-engineering) analysis to identify security and privacy flaws in parental control solutions; for an overview, see Fig. 8.

For each product, we first conducted a dynamic analysis and captured the parental control solution traffic during its usage (as parents/children); if the traffic was in plaintext or decryptable (e.g., via TLS MITM), we also analyze the information sent.

Second, we statically analyzed their binaries (via reverse engineering) and scripts (if available). We paid specific attention to the API requests and URLs present in the code to complement the dynamic analysis. After merging the findings, we looked into the domains contacted and checked the traffic for security flaws (e.g., TLS weaknesses). Third, we tested the security and privacy issues described in Sec. 3.1 against the collected API URLs and requests.

Lastly, in case the parental control solution presented an online interface, we assessed the password-related issues and tested the SSLStrip attack against the login page.

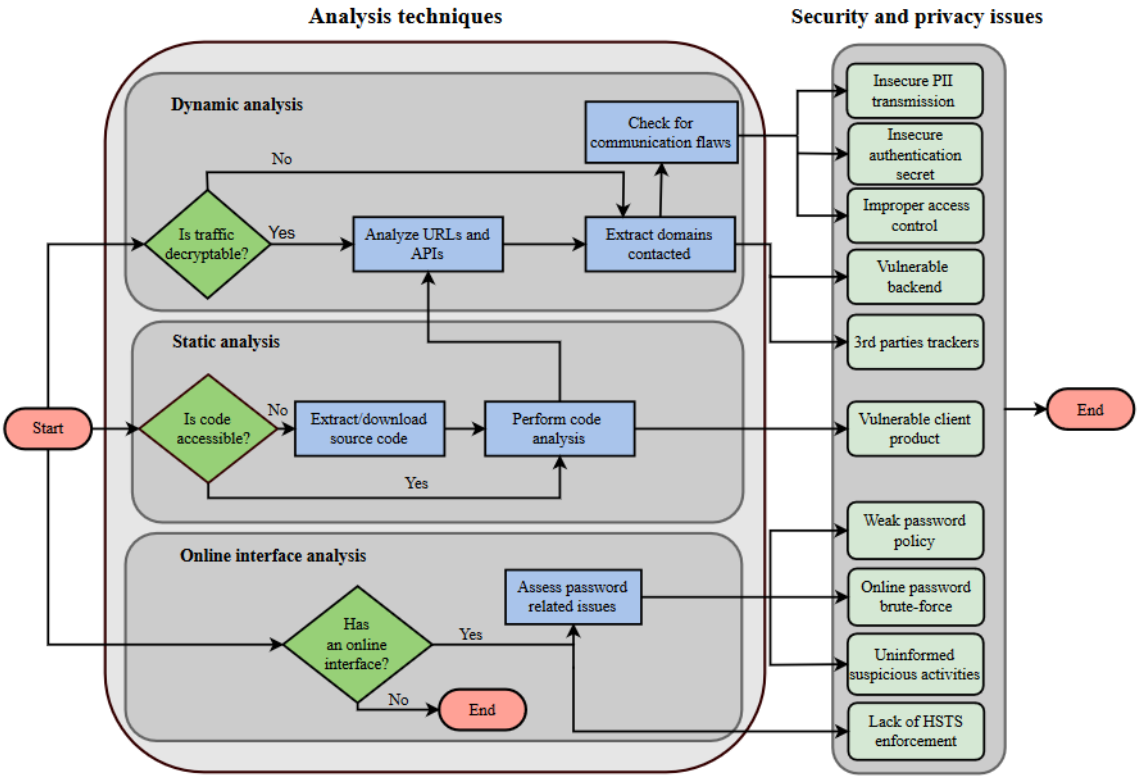


Figure 8: Overview of our evaluation framework.

3.4.1 Dynamic Analysis

We set up test environments for each solution, emulated user actions for hours to days, collected the traffic from the child, parent, and network devices, and then performed relevant analysis (see Sec. 3.1).

Usage Emulation and Experimental Setup. We analyzed each solution by manually mimicking regular users' operations with the goal of triggering parental control mechanisms. We tested for potential vulnerabilities in these mechanisms (see Sec. 3.4.1). For all solutions, we evaluated the web filtering mechanism by visiting a blocked website

(gambling/adult) and a university website. We also perform user activities monitored by platform-specific parental control features (see Sec. 2.1.4, and evaluated the solution's operations. For example, on Android, we performed basic phone activities (SMS, phone call) and internet activities (Instant messaging, social media, browsing, and accessing blocked content).

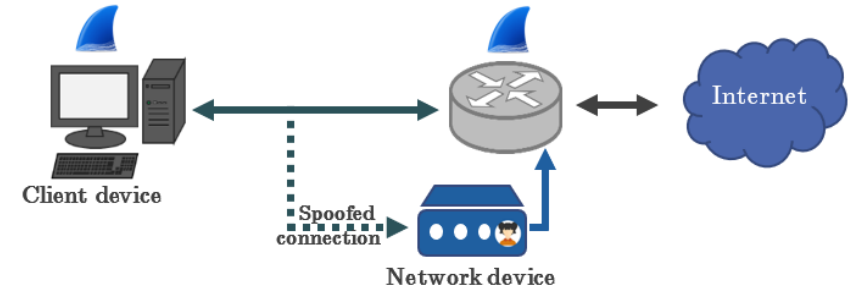
The network devices were evaluated in a lab environment by connecting them to an internet-enabled router like in a domestic network setup. We used a router with the OpenWrt firmware [51], an embedded OS based on Linux, primarily used on network routers. We used test devices with web browsing to emulate a child's device. In case the parental control device used ARP spoofing, the test device was connected directly to the router's wireless access point (AP); see Fig. 9 (a). Otherwise, the test device were connected to the parental control device's wireless AP instead; see Fig. 9 (b). We captured network traffic on both the test device and the router using Wireshark and tcpdump, respectively.

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

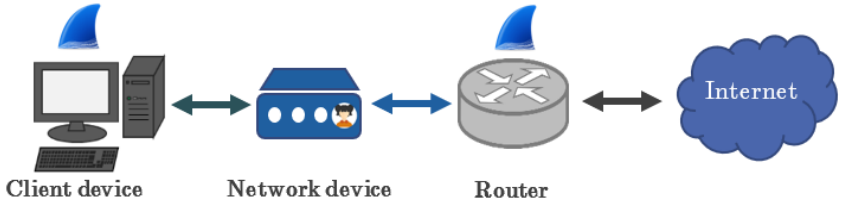
For Android apps, we maintained two experimental environments to concurrently record and inspect network traffic originating from the child and parent apps. We examined the child apps using a One Plus 7 phone running Android 10.0; for the parent apps, we use a Nexus 4 with Android 7. We ran a full Linux distribution with mitmproxy and tcpdump on each experimental environment by installing Linux Deploy [8], and configured Android's network settings to proxy all traffic going through the WiFi adapter to the mitmproxy server. This enables us to capture the network traffic directly within mobile devices.

After intercepting traffic, we parsed and committed the collected tcpdump traffic to an SQLite database and checked for the following security and privacy related issues.

PII and authentication secrets leakage. We examined the collected traffic to check for PII



(a) ARP poisoning case.



(b) Dedicated Access Point case.

Figure 9: Network devices test environment.

and authentication secrets transmitted in plaintext or leakage of PII to third-party domains. We created a list of possible PII that can be leaked via the Request URL, Referer, HTTP Cookie, and LocalStorage. We automatically search for PII items (i.e., case insensitive partial string match) in the collected traffic, and record the leaked information, including the HTTP request URL. We decoded the collected network traffic (including URL, HTTP cookies, requests' payload) using common encodings (base64 and URL encoding) and hashing algorithms (MD5, SHA1, SHA256, and SHA512) to find out obfuscated leaks.

Improper access control. We parsed the traffic to find API endpoints with improper access control. First, we tried to identify all the APIs that can be potentially exploited (without strong authentication), using Postman (`postman.com`) to replay the recorded HTTP request stripped of authentication headers (e.g., cookies and authorization header). Any request successfully replayed were labeled as potentially vulnerable. Afterward, we retrieved the parameters used by these APIs (e.g., keys, tokens, or unique IDs), and assessed the parameters in terms of their predictability and confidentiality. For instance, we deemed

a device's access control insecure if its own MAC address was used for API endpoints authentication, as the MAC address can easily be found by an attacker on the local network.

Identifying third-parties trackers. To identify third-party domains in the parental control solutions traffic, we used the WHOIS [58] registration record to compare the domain owner name to the parental control website owner. In cases where the domain information is protected by the WHOIS privacy policy, we visited the domain's domain to detect any redirect to a parent site; we then lookup the parent site's registration information. If this failed, we manually reviewed the domain's "Organization" in its TLS certificate, if available. Otherwise, we tried to identify the domain owner by searching in `crunchbase.com`.

Backend Assessment. Due to ethical/legal concerns, we refrained from using any invasive vulnerability scanning tools to assess backend servers. Instead, we looked into the backends' software components as disclosed by web servers or frameworks in their HTTP response headers, such as "Server" and "X-Powered-By." We then matched these components against the CVE database to detect known vulnerabilities associated with these versions. Additionally, we used the Qualys SSL Test (Qualys 2020: `ssllabs.com`) to evaluate the security of the SSL configuration of the parental control solutions' backends.

Challenges. During the interception and traffic analysis phase, we encountered several challenges. Those challenges are not unique to our analysis, but they have not been explored for parental control solutions. We summarize them here, including the tools and techniques we use to address them.

Traffic interception. Most network devices used TLS for communicating with their backends and security mechanism prevented us from inserting a root certificate on these devices, so some of the network traffic generated by them is completely opaque to us. In these cases, we rely on static analysis of the device's firmware and differential dynamic analysis to infer behavior. A few studies have been conducted on the topic of monitoring IoT devices from encrypted traffic [15, 80]. However, it remains an open research question.

One Windows application, Qustodio used its own encrypted certificate authorities store to check, accept or reject certificates. Thus we could not regularly intercept the traffic with mitmproxy as it required to install its associated certificate. We extracted the Qustodio TLS proxy private key by dumping its process memory and used its own certificate to replace the mitmproxy one.

Traffic identification. On Android, a key issue is to properly identify the process that generated the traffic in the absence of the packets' referral metadata. We tested how the app behaves when the child uses her device normally (e.g., phone calls, messaging, browsing). These activities produce a large amount of traffic that we need to match to the corresponding processes. We used the mitmproxy addon to call `netstat` to detect the process name for every packet. While we may fail to detect short-lived connections due to the delay of the near real-time netstat. The majority of the connections analyzed consisted of long-lasting connections where netstat allows us to bind connections with PID. We directly used netstat from the underlying Linux kernel (in our Linux Deploy setup) to capture the process ID and process name as soon as a connection is created, while previous work [39, 59] read and parse the system `proc` directory from the Android Linux kernel by checking the directory periodically. This past approach misses connections that are opened and closed before the next time they check the `proc` directory, while our approach looks into the live connection as soon as a connection is created. We may only miss very short-lived connections that are not detected by netstat. To the best of our knowledge, we achieve more reliable traffic-process attribution compared to past work. We leave a full evaluation of the effectiveness of the technique for future work.

3.4.2 Static Analysis

Our static analysis aimed to complement the dynamic analysis whenever we could not decrypt the network traffic (e.g., in cases of network devices using TLS). We used static

analysis to identify PII leakage, contacted domains, weak security measures (e.g., bad input sanitization), or potential flaws in implemented mechanisms.

Network devices. We analyzed the network device firmware whenever possible. We tried three approaches to access the devices' firmware:

- First, we attempted to extract the firmware directly from the device via JTAG, UART, or ICSP interfaces. Sometimes companies remove such debugging and communication interfaces prior to sales to decrease risks of industrial espionage. When interfaces were available, we tried to bypass or brute force the authentication. This method proved to be effective in models running on OpenWrt. We were able to trigger the failsafe mode, change the password before rebooting in normal mode, effectively bypassing the authentication.

- When physical interfaces were removed, we scanned for the presence of open remote admin services (e.g., SSH). However, the devices we scanned had either the port closed or key-protected (while it hindered our analysis, we were glad to see this type of service protected).

- Among the remaining devices without access to their firmware, we tried to download the device firmware from the vendor's website.

We found 3/8 network devices with an accessible serial UART port (KoalaSafe, Blocksi, and Fingbox) that we used to extract the firmware from the devices. Another device, Circle, made its firmware available online.

To identify vulnerable services, we scan the network devices with several tools (OpenVas [50], Nmap [49], Nikto [18] and Routersploit [61]), and match the identified software versions against public vulnerability databases.

Chrome extensions. We manually analyzed the source code of the Chrome extensions. The source code of a Chrome extension mainly consists of scripts, separated into content

scripts and background scripts. Listed in the manifest, the content scripts are injected and ran on pages matching defined patterns or all the pages. Using the Chrome API, the scripts can interact with the page, and query and modify content. On the other hand, background scripts are loaded in the browser's background and are invisible to the user. They aim to sustain the operation of the extension during all the user session activity. As most Chrome extensions' codebase was relatively small and did not involve serious obfuscation, we were able to investigate their operations and detect security and privacy issues (e.g., PII leakage, common JavaScript vulnerabilities).

Android apps. We performed an automated analysis on all 153 Android apps using Firebase Scanner [63] to detect security misconfigurations in Firebase.[1]

We also used LibScout [9] to identify third-party libraries embedded in these apps. Since LibScout does not distinguish which libraries are used for tracking purposes, we use Exodus-Privacy [57] to classify tracking SDKs. We use MOBSF [47] to extract the list of third-party tracking SDKs from all 153 apps based on Exodus-Privacy's tracker list.

3.4.3 Online Interface Analysis

The online user interface is the primary communication channel between parents and parental control solutions. It displays most of the data collected by the solutions, and may remotely enable more intrusive features. Compromising the parent account can be very damaging, and thus we evaluate the security of this interface.

SSLStrip attack. To check for SSLStrip attacks, we first set up a WiFi AP with mitmproxy [21], SSLStrip2 [38] and Wireshark [77] installed. Then, we connect the parental control solution to our WiFi access point. Wireshark is utilized to record network traffic while mimicking common use case scenarios with the goal of triggering all parental control

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

monitoring and control UI and API requests looking for signs of successfully SSL Stripping attack on the traffic. We confirm the effectiveness of the attack by comparing the result to the corresponding traffic in a regular testing environment (i.e., without SSLStrip).

Weak password policy. During the parental control solution's account creation, we evaluate its password policy. We adopt a fairly conservative stance and only labeled as weak the password policy accepting password with 4 characters or less. We also evaluated the presence of most advanced requirements: usage of number, special char, lower and upper case char. Finally, we tried to use password from the top 10 passwords used (and highly insecure).

Online password brute-force. We use Burp Suite [56] to perform password brute-force attacks on our own online accounts. To keep the load on the server minimal, we test for the presence of defensive mechanisms by 50 attempts on our account from a single computer.

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

Chapter 4

Results

In this chapter, we present the results of the analysis of 41 parental control solutions across four different platforms. We identify 92 vulnerabilities and reveal that the majority of the analyzed parental control solutions threaten the child privacy and security. As each platform has its own unique characteristics, monitoring capabilities and restrictions, we group the results by platforms and vulnerability types. An overview of the results is provided in Table 7.

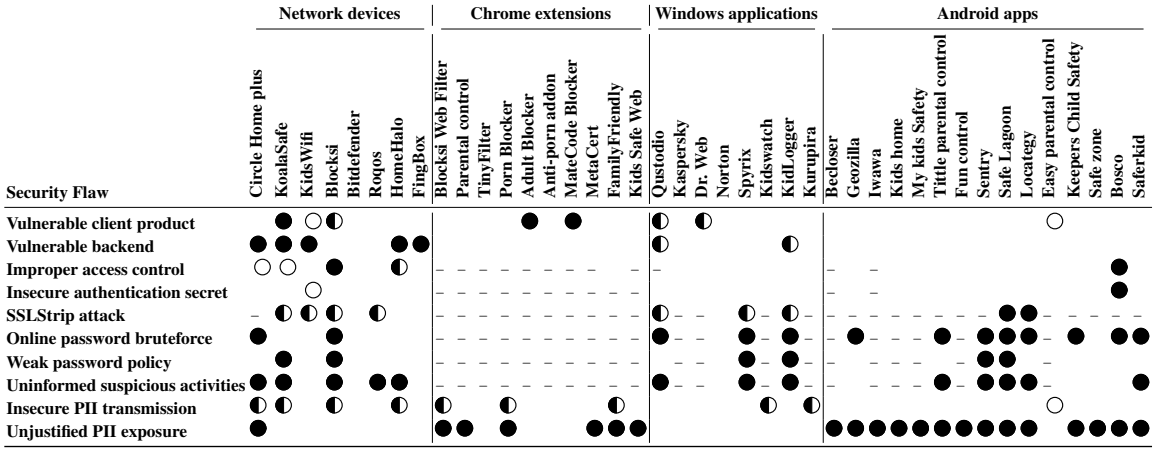


Table 7: Overall results for security and privacy flaws in parental control solutions labeled following the threat model in Sec. 3.2. ○ : Local network attacker; ◐ : On-path attacker; ● : Remote attacker; −: not applicable; `blank`: no flaw found. In case the vulnerability can be exploited by two types of attacker, we display the fullest circle applicable.

4.1 Windows Applications

The Windows application analysis preceded the formal implementation of the methodology described in Sec. 3.4. It integrated tests related to the specificity of the monitoring features on Windows (mainly TLS proxy). Some of the tests carried out were deemed as successful, and served as a basis for the methodology subsequently developed, while others proved to be less relevance and were not included.

4.1.1 Analysis of the TLS-proxy

One of those Windows-specific tests concerns the TLS-interception feature used by parental control Windows applications. Our initial security analysis drew inspiration from the research conducted by X. de Carné de Carnavalet and M. Mannan [22] on TLS-interception proxy. First, we wanted to assess the current state of the implementation of the proxies in parental control applications with regards to the previous observations. TLS interception involves the delegation of security checks from the browser to the proxy and thus carries an important security implication. We performed four types of security checks:

- *Certificate generation:* Modern web connections rely on TLS to guarantee basic web security. Certificates are mainly used to guarantee the identities and prevent malicious actors to impersonate known domains. In the context of a TLS-proxy, the content received by the web browser is first certified by the proxy certificate. This proxy certificate is beforehand trusted by either the web browser or by the OS (included in the Trusted Root Certification Authorities Certificate Store). Therefore, the generation of the proxy certificate and the confidentiality of its private part are critical. If the secret key is leaked or shared across devices (e.g., statically generated or guessable) it is possible for an attacker to make the web browser trust malicious content. To determine whether a proxy certificate was dynamically generated we

compare the private key of multiple generated root certificates of the same parental control solution. Those certificates are generated by going through the complete installation procedure on different clean instances of the same virtual environment multiple times.

- *Certificate validation:* In a TLS-interception situation, the proxy also replaces the web browser in its role of communication with the servers. Therefore, a correct validation procedure of the received certificates by the proxy is a crucial step, especially to detect faulty or forged certificates. Previous studies[22] showed that this validation was poorly operated by some applications and consequently accepting faulty certificates and putting the users at risk. We assess certificate validation using online tools, `badssl.com` and `ssllab.com` and completed those tests by crafting special faulty certificates and connecting the machine with the proxy to a local domain using those certificates.

- *TLS parameters:* In addition to the security checks involving certificates, it is important to ensure that the secure connection parameters are neither outdated nor vulnerable. The security provided by the proxy should not be any lower than the security of an updated web browser. Otherwise, it would create an illusive sense of security for the user. For this purpose, we established connections between the proxy and a server using different protocols, unsecured (e.g., SSL3, TLS 1.0) and secure (e.g., TLS 1.2, TLS 1.3). We also assessed the cipher suites supported and accepted by the proxy.

- *Know practical attacks:* We also tested these proxies against prominent TLS-based attacks. We tested FREAK, Logjam and CRIME.

4.1.2 Initial Dynamic Analysis

The second analysis performed on Windows applications concerns the study of data sharing mechanisms. Through previous work and our preliminary study we distinguished two data sharing behavior: (1) triggered and (2) periodic communications. Triggered communication refers to any data sharing triggered by a user action (e.g., software use, search engine query or website connection). On the other hand, periodic communication involves data sharing at a given frequency and independent of the user action.

To identify data shared through periodic connection, we opted to record network connections over an extended period. The objective was to retrieve information about the connections issued by the software (e.g., domains contacted, frequency, and content). We implemented two types of monitoring, passive with Wireshark and Sysmon, and active with mitmproxy. The passive monitoring allowed us to identify the connections issue by software, their frequencies, the quantity of data shared and the domains contacted. The active monitoring relied on MITM proxy to intercept secure connections to extract the content of the communication. The recording time for the passive monitoring was of 3-5 days (depending on the application) while the active method spanned for 24 hours, see Table 8 .

We used Selenium paired with the Windows task manager to automate action on the Chrome web browser at regular intervals. The web crawler visited the top 5 google results for each search by repeating the procedure every 5 minutes for a day. The purpose of this test was to detect an action pattern in the proxy suggesting an irregular data sharing coinciding with the search done. An iteration of the test was also conducted using a mixed URL list composed of legitimate domains from Alexa top 50 for USA (filtering adults website) and unsuitable domains from a blacklist maintained by the University of Toulouse, France [70].

One of the objectives of those tests was to detect odd pattern (e.g., data sharing with

	Frequency of communication	Type of communication	Encrypted channels
Qustodio	5mins	API	✓
Kaspersky	20mins	API	✓
Dr. Web	5mins	API	✓
Norton	10mins	API	✓
Spyrix	1.5min	API	✓
Kidswatch	1 day	API	
KidLogger	2.5mins	API	✓
Kurupira	no default value	email	✓

Table 8: Windows parental control application-server communications.

a third party) when connecting to websites of different types. Connections made by the solution to domains unrelated to its backend would be an evidence of advertising, tracking or other undesirable activities performed by the parental control solution. To narrow the scope of the analysis and identify these connections to third-party domains with a better precision, we set up another experiment. For each Windows parental control solution, we visited multiple times a list of several websites: social media, academic domain, governmental domain, and media. We automatically parsed and compared the network records with a control recorded in similar conditions (but without a parental control solution installed). To avoid false positives, we considered a connection to a domain as issued by the solution when we found occurrences of it in two-third of the visits for a given URL.

We did not generalize the aforementioned tests to the other type of parental control solutions. From a result-oriented perspective, these experiments did not reveal the presence of any third-party trackers. Second, our understanding of the parental control solution evolved during the analysis. We changed our assumption of parental control solutions from potential rogue malware to probable flawed applications. Hence, we accordingly modified our analysis to the current form described in Sec. 3.4.

4.1.3 Security

Vulnerable client product. On the bright side, critical vulnerabilities pointed out by previous work on parental control solution's TLS proxy have been corrected, especially regarding the generation of the certification. However, we still found some weaknesses, in particular in the certificate validation. Other than Kidswatch, all tested Windows applications relied on TLS-interception proxies to operate. We found that some of these proxies do not properly perform the certificate validation step. Qustodio and Dr. Web accepted intermediate certificates signed with SHA1, despite recent research enhanced collision attack on SHA1 [40]. Dr. Web also accepted Diffie-Hellman 1024 considered weak [1], and deprecated in Safari and Chrome since 2016 [16]. In addition, none of the proxies analyzed rejected revoked certificates. We also found that upon uninstallation of the applications, the root certificate associated with the proxy remained in the Windows trusted root certificate store, with four of them having a validity duration over one year.

Vulnerable backend. We found that some Windows applications' servers do not always use ideal TLS configurations. For instance, Qustodio server's certificate chain of trust contains an intermediate certificate signed with SHA1. Furthermore, Qustodio and KidLogger servers support and issue communications using the RSA key exchange protocol, which lacks forward secrecy.

SSLStrip attack. We found all the Windows applications with an online interface except Norton are vulnerable to SSLStrip attack (Qustodio, Spyrix and KidLogger). They transmitted credentials in plaintext under an SSLStrip attack. This allows an adversary to compromise the parent account for a long time, particularly if the app does not send any notification to the parent when the account is accessed from a new device (see uninformed suspicious activities).

Password policy. Spyrix and KidLogger enforced a *weak* password policy (password shorter than four character).

Online password brute-force. All the Windows applications with an online interface except Norton fails to protect their login interface against online password brute-force. We were able to continuously attempt to login (>50 tries) to the same account without triggering any defense mechanism (e.g., timeout, captcha or IP ban).

Uninformed suspicious activities. All three applications did not report suspicious activities performed on the parent's account (password changes, access from unrecognized devices). These activities are possible indicators of account compromise and should be reported to the user.

4.1.4 Privacy

Insecure PII transmission. Kidswatch is probably the most problematic Windows parental control application among the ones we analyzed. This application is no longer advertised in the top ranking for 2018-2019 but is still displayed by some websites and was part of the 2017 and anterior years top rankings. However, this application was found to be defective on several levels. Among the Windows applications, it is the only one that sends unencrypted traffic to its server. The application sent reports in the form of an email via HTTP to `kidwatch.com/kw50/mail/mail/mail_addqueue.php`. On a different note, 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's SMTP server uses an unencrypted protocol, Kurupira does not warn about the risk involved in the transmission of child activity reports in plaintext.

Unjustified PII exposure. Following the disclosure of the privacy flaw in Kaspersky application [25] described in Sec. 2.2, we conducted a static analysis on the other Windows applications to determine whether they added scripts through the processing of the TLS proxy and if these scripts included identifiable elements susceptible to be used to track users. We found that only the Kaspersky parental control application presented this type of

behavior. In addition, and in contrast to solutions on other platforms, we did not observe any communication between Windows solutions and third parties.

4.2 Network Devices

This section details the results of analyzing the eight network devices. This study was conducted in cooperation with another researcher. We purchased and manually analyzed these devices between September 2019 and May 2020. Overall, we found vulnerabilities in all but one device (Bitdefender Box).

4.2.1 Security

Vulnerable client product. The importance of securing the update mechanism has been known for years, cf. [11]. Surprisingly, the Blocksi firmware update happens fully through HTTP. An integrity check is done on the downloaded binary image, using an unkeyed SHA256 hash, again retrieved using HTTP, and thus rendering it useless. Therefore, an on-path attacker can trivially alter the update file and inject their own malicious firmware into the device. We confirmed this vulnerability to be exploitable by performing the attack on our device.

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

In addition, we noticed that KoalaSafe runs Dropbear v2014.63 SSH server/client (released on Feb. 19, 2014), associated with four known remote code execution vulnerabilities. Under certain conditions, the KoalaSafe device opens a reverse SSH tunnel through its backend server, exposing the vulnerable SSH Dropbear server to an attacker outside the local network. By calling a KoalaSafe API endpoint (`https://api.koalasafe.com/api/router/[MACaddress]/et`) an external attacker can detect when a reverse SSH tunnel is open using only the victim device's MAC address. If the tunnel is open, the API endpoint responds with the tunnel's port number, 0 otherwise. The same API is used daily by the device to know if he should open the developer reverse tunnel. For large-scale exploitation, an attacker can query the aforementioned API endpoint to enumerate all KoalaSafe devices with the reverse tunnel open. This enumeration is feasible as KoalaSafe uses the GuangLia network interface card (NIC), and MAC addresses assigned to GuangLia NICs [67] are limited to only 2^{20} values.

Vulnerable backend. The results of our analysis of the backend server API endpoints are summarized in Table 9. 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 three known CVEs. Interestingly, all of the backend servers using Nginx are running old versions that are vulnerable to the same three CVE's which impact the server's availability. The Blocksi's API backend only indicates that it runs on OpenResty and Google Frontend. However, 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.

Improper access control. The KoalaSafe API login endpoint requires three parameters

Device	Software Components	# of CVEs
KoalaSafe	Apache 2.4.34	11
Circle	Nginx 1.15.5	3
KidsWifi	Nginx 1.10.2 PHP 7.0.27	3 26
HomeHalo	Nginx 1.12.0	3
FingBox	Nginx 1.12.2	3
Blocksi	OpenResty, Google Frontend	N/A
Bitdefender	N/A	N/A
Roqos	N/A	N/A

Table 9: Vulnerable software components on backend APIs. *N/A*: Not enough information could be extracted.

that are available to anyone on the local network: a device-generated authentication token, the device's date and time, and the device's MAC address for successful authentication. These parameters can be obtained by visiting endpoints hosted by the KoalaSafe device. The authentication token and device time are available at `https://device.koalasafe.com/auth.lua`, and the MAC address at `https://device.koalasafe.com/status.lua`. Thus, a local network attacker can easily collect the information needed for authentication and use the API endpoint to access sensitive information such as the profile name, email address, and browsing history.

For Blocksi's login API endpoint, the device's serial number (SN) and the registered user's email are required to authenticate the device to the server. However, a remote attacker needs to know only one of these parameters to authenticate. This is because a remote attacker can retrieve a user's email using their device SN or vice-versa; For SN to email, use `https://service.block.si/config_router_v2/router_checkRouters/null/[SN]`, and for email to SN, `https://service.block.si/config_router_v2/router_checkRouters/[email]`. By sending both parameters to the API endpoint in a POST message, any remote attacker can authenticate to the server, and access sensitive information about the home network (e.g., the WiFi password, and MAC addresses of connected devices).

The HomeHalo device uses only the device's SN and an HTTP header called `secret Token` to authenticate to its API endpoint. In our case, the `secretToken` had a fixed value of 100500. Although we haven't tested this hypothesis on multiple HomeHalo devices, the `secretToken` doesn't appear to be randomly generated and could either be constant across all devices or enumerable. An on-path attacker can intercept and modify these messages, and gain access to admin controls (e.g., reading or changing the wireless SSID, password, or even the device's root password). Other privacy sensitive information is also exposed, including the devices connected to HomeHalo's network and the parental control profile setup.

The Circle Home Plus creates a profile for each child and stores it locally on the device, including the child age groups, usage history and statistics, child photo, and username (i.e., some parents may use the real child name as username). We identify two API endpoints used to transmit child information over the local network. The first API endpoints, `http://[CircleIP]/api/USERINFO?host=ios&nocache=1572292313630HTTP/1.1`, sends child account usage history and statistics, and `profileID`. It insecurely relies on the requester's MAC address to identify the child device and communicate sensitive information. This API endpoint is called whenever a child device attempts to access a restricted domain. The second API endpoint, `http://[CircleIP]/api/USERPHOTO?profileID=[profileID]`, fetches the profile photo corresponding to the received profile ID. 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 leak sensitive information being exchanged over the network. Alternatively, a local attacker who can spoof her MAC address can easily extract this information. Thirdly, a malicious app or SDK installed on the child's device can also launch this attack. We confirmed this scenario by creating a dummy Android app. This app didn't request any

additional permission and could call the APIs, collect information and sent it back to its server in a completely transparent way for the user.

Insecure authentication secrets. 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.

When launched, Blocksi creates kid and parent wireless networks. These networks are set up such that they share a default password common to all the Blocksi devices. Although Blocksi recommends changing the password as part of the setup, it's not mandatory and we can expect that a significant number of users would skip this step [12]. Thus, if the owner of the device does not change this default password during the setup, an outsider can connect to either network using these default credentials.

The KidsWifi user web interface generates a token when a user connects to it. This token is used as 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 [28]. Local storage does not have the cookie `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. Which means that any regeneration of the token will give the same. 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.

SSLStrip attack. We found that four network devices, KoalaSafe, Blocksi, Kidswifi and

Roqos are vulnerable to SSLStrip attack. They transmitted the parent account credentials via HTTP under an SSLStrip attack. This allows an adversary to compromise the parent account for a long time, particularly if the app does not send any notification to the parent when the account is accessed from a new device (uninformed suspicious activities).

Password policy. Two network devices, KoalaSafe and Blocksi, enforced a *weak* password policy (password shorter than four characters). In fact, both KoalaSafe and Blocksi simply do not enforce any password policy, accepting any length and any type of characters as a password. Concerning the others, the requirements when choosing a password are more up-to-date. The chosen password must contain a certain amount of characters, a combination of number, lower and upper case chars. However, no blacklists are applied and common and weak passwords such as “Password123" or “Qwerty1234" are accepted despite their lack of robustness. Regarding Roqos, we observed the presence of a password strength meter to assist the user when modifying the password. It incorporates a dictionary and warns the user of a weakness in the password when recognizing words. For example, passwords containing “password" or “admin" trigger a “worst" rating by the password strength meter. However, the password strength meter does not impose any constraints, very weak and “worst" passwords are accepted the same way an “excellent" password would be. In the end, the only restriction enforced is the presence of at least 8 characters.

Online password brute-force. We found that two network devices, Circle and Blocksi, leave their online login interface open to password brute-force attacks. In essence, we were able to attempt multiple logins for the same account without triggering any forms of defense mechanism (e.g., timeout, captcha or IP ban). This vulnerability is worsened when the online interface has a bad password policy allowing the use of very weak password, shorter and less complex, thus easier to break by brute force. All the other network devices make the choice of timeout after a certain amount of attempt, while this solution is not perfect it does hinder simple password brute-force attempts.

Uninformed suspicious activities. We noticed that five network devices, do not report suspicious activities on the parent's account such as password changes and accesses from unrecognized devices. These activities are possible indicators of account compromise and should be reported to the user.

4.2.2 Privacy

Insecure PII transmission. 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. By binding the MAC address to the DNS request, these products keep track of the child devices' history and enforce the profile-based restrictions. However, regular DNS requests are not encrypted. Therefore, by adding the user MAC address to the request, the device makes the user activity traceable by on-path attackers (cf. [20]). The HomeHalo device suffers from a similar problem: whenever a domain is requested by a user device inside its network, HomeHalo sends an HTTP request, including the child device's MAC address, to its backend server to identify the requested domain's category.

Unjustified PII exposure. We evaluated potential uses of third-party tracking SDKs in the parental control solutions. For network devices, we identified the use of third-party SDKs in the companion apps but not in the firmware. Our static analysis for five companion apps (three devices does not have it) reveals the use of tracking SDKs (2–12 unique trackers) in four on them. Effectively, Fingbox companion app presented twelve, Circle six trackers, Roqos three, Bitdefender two and KoalaSafe zero.

From the dynamic analysis, we also found that one of the network devices' companion app, Circle, includes a third-party analytical SDK from Kochava. Every time the app is launched, or it returns to the foreground, the following information is shared with Kochava: Device ID (enables tracking across apps), device data (enables device fingerprinting for persistent tracking). Kochava provides an opt-out option (`app_limit_tracking=true`)

that can be used to comply with COPPA. However, the app transmits this flag as `false` from the child device. Note that Disney, a former partner of Circle (involved in the 1st generation of the circle product), is the target of a class action lawsuit for using a similar SDK in children's apps [71].

4.3 Browser Extensions

This section presents the result of the analysis of ten Chrome extension. The analysis was conducted from April 2020 to June 2020. Extensions have certain peculiarities due to their nature, notably the absence of an online parental interface. From a security perspective, the static controls implemented by Chrome detect and filter out problematic extensions. As soon as a new form of vulnerability is brought to light by security experts, Chrome controls and disables hundreds of vulnerable extensions. Our analysis, therefore, focused mainly on privacy issues, also emphasized by several articles and research (see Sec. 2.2).

4.3.1 Security

Vulnerable client product. Two Chrome extensions (Adult Blocker and MateCode Blocker) download and run a third-party tracking script at run time (launched by the background script). The domains hosting the scripts are apparently neither related to the extension providers nor well-known libraries: `dc.airdropanalytics.com` and `thatheme.club`. This type of operation raises two major concerns:

- Runtime loaded scripts bypass the static control of Chrome for extension security, which has been exploited in the wild by tricking developers into adding malicious scripts masquerading as tracking scripts [34].
- The surface of attack includes both the extension developer and the third party, doubling the risk of malicious takeovers or phishing attacks.

4.3.2 Privacy

Insecure PII transmission. We found that three extensions, Blocksi Web Filter, FamilyFriendly Parental Control and Porn Blocker transmitted the domains contacted by the user in plaintext. The two first extensions sent the domain name requested by the user to their server to check whether the website contacted is suitable for children. The communication happened fully over HTTP. On the other hand, Porn Blocker leaked the full URL of the current page when trying to connect to a blocked domain. The extension redirects the user to `https://www.purplestats.com/page/blocked/` when visiting a blocked website. Indeed, the HTTP `referer` request header contains the full URL of the page from where the connection is issued. The Referer header leakage is a known issue in the web extension privacy analysis field, particularly regarding the use of custom toolbars, which are requested and injected in each page (leaking the full URL by the same way).

Unjustified PII exposure. The majority of the Chrome extensions send browsing information to their server. Four extensions sent the requested domains to their server to check whether the website should be blocked. Although this would appear as a legitimate operation, a better approach would be using a local database in the client browser. Google safe browsing in its earliest version acted similarly to the applications studied, by sending the complete URL. In response to privacy concerns, it opted to perform with an imported database and send anonymous data (truncated hashes) to the servers instead (note that this practice still raises privacy concerns [32]). Using a local database and check would permit to avoid unnecessary privacy concerns introduced by PII sending [35]. Two more concerning extensions sent the complete URL, possibly leaking personal information. Furthermore, the full URL is irrelevant to determine whether the website is suitable for children or not.

Another extension, Parental Control, overrode Chrome setting and replaced the default search URL by its server domain, which automatically redirected to Google Safe Search. Similarly, the information sent to the extension's server can be avoided and constitutes a

privacy leakage.

On a different level, half of the extensions analyzed did not have a privacy policy or were unavailable. While we could argue that the stricter User data policy Chrome enforced starting Fall 2019 (project Strobe [17]) may not apply to the parental control extension analyzed. The user is therefore not informed of the usage done of the information sent back to the extension server (e.g., URL, search terms, browser information, location information).

Extension	Data leaked	Analytical service contacted	Privacy policy available	Analysis result
Blocksi Web Filter	Domain		Yes	Privacy leakage
Parental control	Domain	Google-Analytics, analytics.vmn.net	No	Privacy leakage
TinyFilter		Google-Analytics, doubleclick.net	No	Suspect
Parental Control: Porn Blocker	URL	purplestats.com	Yes	Privacy leakage
Adult Blocker- Porn Adult Filter			Yes	Suspect*
Parental Control - Adult Blocker		Google-Analytics	Yes	Safe
Adult Website Blocker | Porn Blocker			No	Suspect*
MetaCert - Adult Content Blocker	URL		Yes	Privacy leakage
FamilyFriendly	Domain		No	Privacy leakage
Kids Safe Web	Domain	Google-Analytics	No	Privacy leakage

Table 10: Chrome extensions privacy analysis results. *Suspect**: The extension downloads a script from a third party at the beginning of each user session and extension operation is susceptible to be unexpectedly modified.

We report a conservative rate of 60% privacy-leaking and 90% including suspected extensions. The differences between these figures and those reported by previous studies (between 2% and 7%) can mainly be explained by the type of extension and the scope of the analysis. Many parental control extensions send personal data such as the URL as part of their operations. In terms of content leaked, the parental control extensions do not differ from the results reported in studies with a broader scope, mainly exposing browsing data (domain and URL). For comparison, we have subjected our list of extensions to the tool developed and used by a previous work: Mystique [14]. From the 6/10 extensions that we formally identified as privacy leaking, only two were correctly identified as privacy leaking by the tools. On the other hand, the tools did not reveal any leaks that we were unaware of. This comparison doesn't mean to undermine the merit of Mystique, which due to its automation and its scalability permits the analysis of a very large number of extensions

in a reduced time, but rather to emphasize the value of manual analysis as a supplement, especially when analyzing a specific topic.

4.4 Android Solutions

This section describes the result of the analysis of 15 Android solutions (child and parents apps). This analysis is the second set of analysis on Android parental control solutions using the methodology presented in Sec. 3.4, the first one was conducted by another student and thus will not be discussed in this section. This study was conducted from July 2020 to September 2020.

4.4.1 Security

Vulnerable client product. We found that the application Easy parental control threatens the security of the child device and its owner on various levels. The application is designed to be used by the child and parent on the same local network. However, all the communications issued by the child app insecurely occur over HTTP, e.g., `http://[ChildIP]/pair?p=[pairnumber]&c=[lock/sendMsg&v=message...]`. Both the child IP and the pair number can easily be eavesdropped. This flaw has for consequence to allow a local attacker (e.g., a physical person with the WiFi access or an app installed on the child / parent phone) to perform different operations on the child's phone. The attacker can extract the child apps, lock the device (soft locking) or send direct messages to the child via the application. The latest feature can lead to physical harm as the attacker can impersonate the parent and lure the child outside.

Improper access control. The application Bosco communicates with its backend via

POST requests. The requests are protected by Json Web Tokens, dynamically generated for each user. However, the endpoint failed to check the relation between the authorization token provided and the information requested. Any parent with a valid token can request information about any registered child knowing only its ID. For example, a remote attacker can get the child position using `https://production.boscoserver.com/batteryapigetChildBatteryAndLocation` with the body data in Fig. 10. Alternatively, using `https://production.boscoserver.com/gateApi/getContentInfo` parameters and similar body information the attacker has access to all the content labeled as unappropriated used/ visited by the child. Without any authorization token, it is also possible to request all the parent information associated with the posted child ID, including the parent unique ID, email, name and picture (if present). Lastly, we were able to get the phone call history using `https://production.boscoserver.com/voiceapigetCallsList`.

```
childUUID:[Child ID]
clientVersion:75.41
protocolVersion:4.0
platform:android
clientType:parent
clientOS:android
fullClientOs:
deviceModel:
manufacturer:
languageCode:
operator:
operatorMCCMNC:
partnerId:global/bosco
parentUUID:[not checked by the backend]
developerId:
sandboxText:
```

Figure 10: Body data used to get child information using Bosco API.

Insecure authentication secret. The issue discussed above is worsened by the usage of the

Android advertising ID (AAID) as unique user identifier by Bosco. The AAID is a unique personal semi-persistent Identifier. It is generated by each android phone to allows developers and marketers to track activity for advertising purposes. This ID can be requested by any apps installed on the user phone and is the primary identifier used for marketing. Any android user can manually reset it via the phone setting, however most users are oblivious of the ID usage. Issues occur when this ID is used as a means of identification by Android app API endpoints. The scale of the aforementioned improper access control would be much less important if the unique child identifiers were randomly generated string, without the possibility to link a specific user to a known ID.

SSLStrip attack. Only 3/15 of the Android solutions tested provided an online interface and two were found vulnerable to SSLStrip attack. SafeLagoon and Locategy online interface transmitted parent account credentials via HTTP under an SSLStrip attack. This allows an adversary to compromise the parent account for a long time, particularly if the app does not send any notification to the parent when the account is accessed from a new device (uninformed suspicious activities).

Password policy. Sentry and Safe Lagoon enforced a *weak* password policy (password shorter than four characters). We also noticed that Safe Lagoon gives contrary information about the password policy, asking for strong passwords (alphanumerical and symbol 6+ characters long password) but still accepted 4 characters and less long passwords.

Online password brute-force. All but one Android solution implementing a login system do not protect their login interface against online password brute-force. We were able to continuously attempt to login (>50 tries) to the same account without triggering any defense mechanism (e.g., timeout, captcha or IP ban). Interestingly, we found that a few applications (e.g., Safe zone) delegate their login system to Google by asking the user to login with a Google account linked with the phone. We also noticed that some applications used imaginative captcha alternatives during the account creation (e.g., Iwawa) but

unfortunately not in the login process.

Uninformed suspicious activities. Five android solutions did not report suspicious activities performed on the parent's account (e.g., password changes, access from unrecognized devices). These activities are possible indicators of account compromise and should be reported to the user.

4.4.2 Privacy

Insecure PII transmission. Easy parental control Solution communication between the child and the parent happens over HTTP. The applications installed and the phone usage are leaked over the local network and any device and application connected to the network can eavesdrop the information.

Unjustified PII exposure. Most of the analyzed Android solutions implemented one or many third-party trackers. Through the static analysis using Libscout, we found that 41/51 of the children application (installed on the child phone, 22/24 of the parent application (installed on the parent phone) and 73/78 of the shared application (installed on both type) included trackers libraries see Table. 11.

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

Table 11: Use of tracking SDK in Android parental control applications found through static analysis.

More than 25% of the children apps utilized well-known advertisement networks (Doubleclick, Google Ads), see Fig. 11.

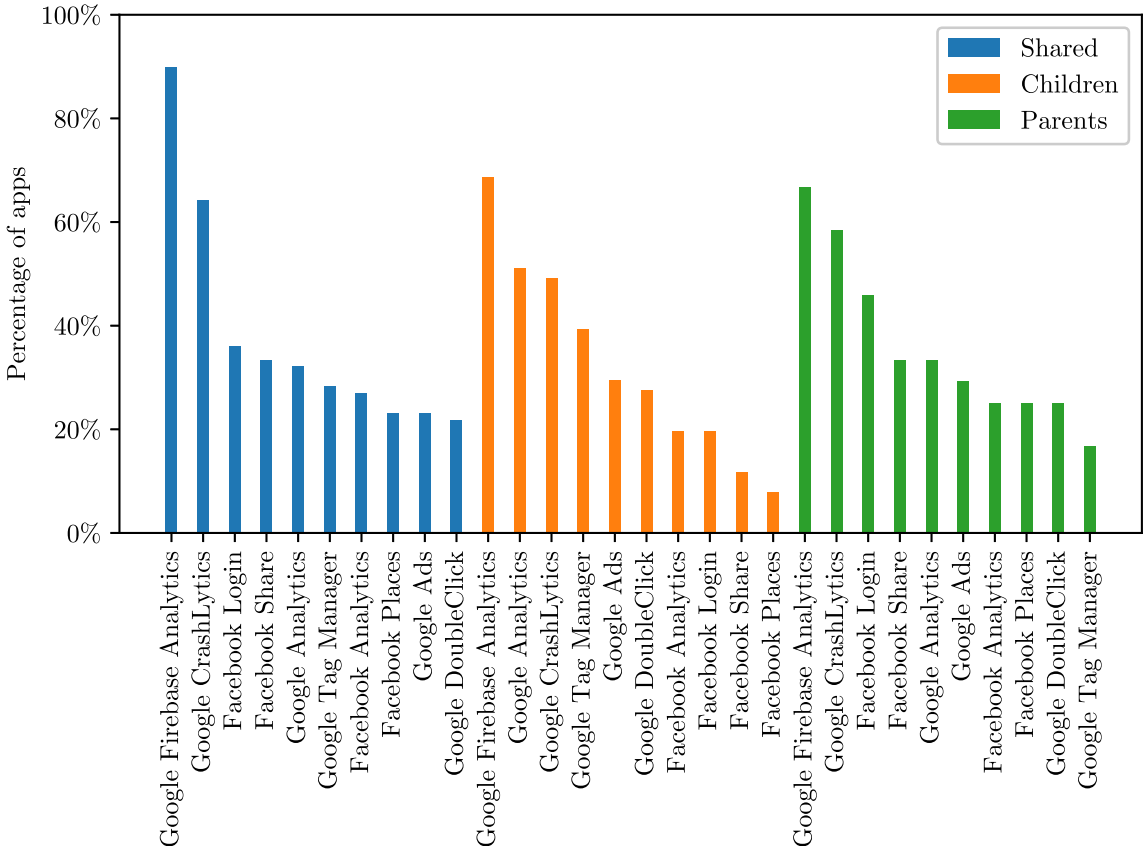


Figure 11: Tracking SDKs present in Android apps found through static analysis.

We also identified the third-party trackers installed on the child phone from the network traffic generated by the use of the tested applications during the dynamic analysis. We found that, except Easy parental control, all the Android solutions installed on the child device communicated with at least one third party, see Fig. 12.

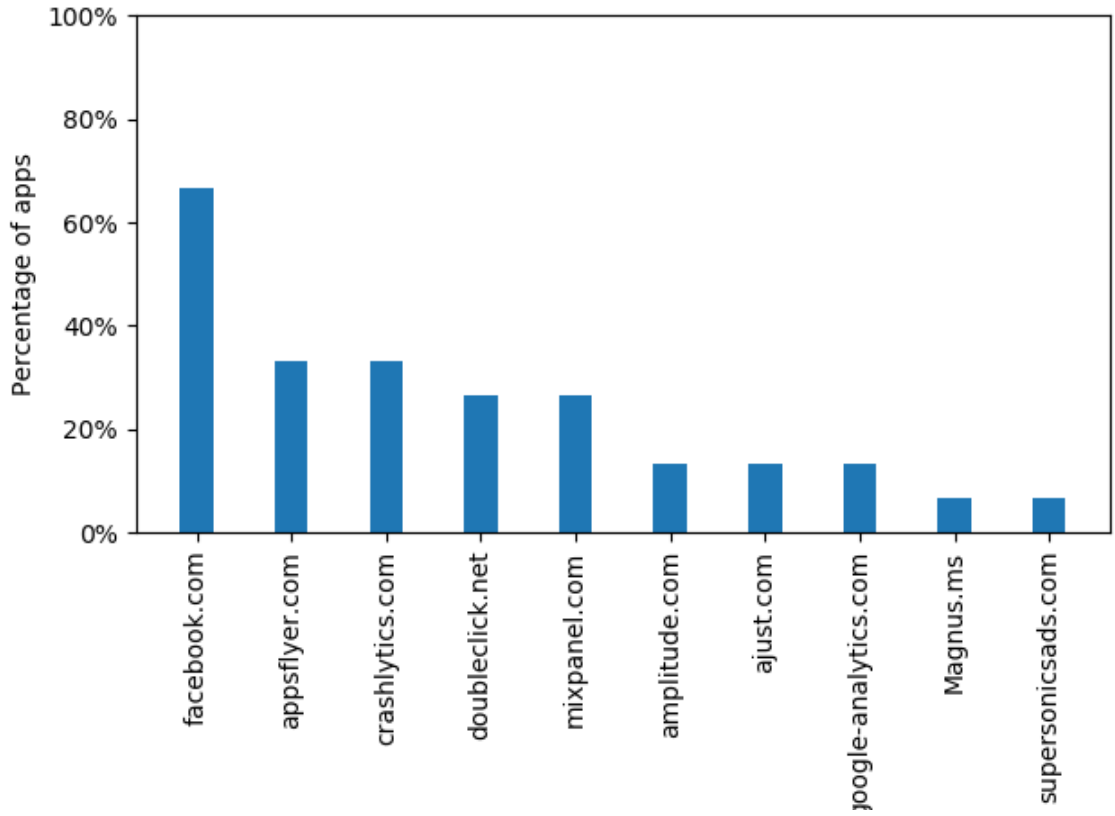


Figure 12: Tracking SDKs present in Android apps found through dynamic analysis.

The dynamic analysis confirmed that several applications did not comply with the regulations in force regarding the protection of children's data. The Coppa compliant parameters, restricting tracking, were not or hardly used by the applications when they were available. As a result, 70% of the applications tested send data to graph.facebook (the facebook analytics platform) with `advertiser_tracking_enabled=true`. 100% of the applications using ajust.com supplied information with the `tracking_enabled=1` parameter. In order to comply with the legislation reinforcing the protection of minors, several analytics service companies have decided to restrict or prohibit their service to applications aimed at children. Among the third parties detected during dynamic analysis,

Crashlytics and Amplitude forbid developers to embed their SDK in children's apps. However, 6/15 android solutions children apps used analytics solutions stating in their privacy policy that their service should not be used for children's apps. The Android apps analyzed shared numerous PII with third parties, see Table 12.

Table 12: Android apps sharing PII with third parties.

Solution	Shared PII	Third parties (number, domains [max. 2]) ⋆
Becloser	AppID	1 (magnus.ms)
Becloser	AAID	2 (magnus.ms, ajust.com)
Becloser	GSF ID	1 (ajust.com)
Becloser	Mobile carrier	1 (supersonicads.com)
Geozilla	AAID	5 (branch.io, localytics.com)
Geozilla	Mobile carrier	5 (branch.io, facebook.com)
Iwawa	AAID	2 (facebook.com, mixpanel.com)
Iwawa	AppID	1 (mixpanel.com)
Iwawa	Mobile carrier	1 (doubleclick.net)
Kids home	AAID	1 (mixpanel.com)
Kids home	Mobile carrier	1 (doubleclick.net)
My Kids Safety	AAID	4 (amplitude.com, appsflyer.com)
My Kids Safety	Mobile carrier	4 (amplitude.com, appflyer.com)
My Kids Safety	Geoposition	1 (onesignal.com)
Tittle parental control	Mobile carrier	1 (doubleclick.net)
Tittle parental control	AAID	1 (facebook.com)
Tittle parental control	Mobile carrier	1 (facebook.com)
Funcontrol	Mobile carrier	2 (doubleclick.net, facebook.com)

Continued on next page

Table 12 – continued from previous page

Solution	Shared PII	Third parties (number, domains [max. 2]) ⋆
Funcontrol	AAID	2 (facebook.com, mixpanel.com)
Funcontrol	AppID	1 (mixpanel.com)
Sentry	AAID	1 (appflyer.com)
Sentry	Mobile carrier	1(appsflyer.com)
Locategy	AAID	1 (facebook.com)
Locategy	Mobile carrier	1 (facebook.com)
Keepers	AAID	3 (appflyer.com, facebook.com)
Keepers	Mobile carrier	3 (appsflyer.com, mixpanel.com)
Safe zone	AAID	1 (facebook.com)
Safe zone	Mobile carrrier	1 (facebook.com)
Bosco	AAID	2 (facebook.com, appsflyer.com)
Bosco	Mobile carrier	2 (facebook.com, appsflyer.com)
Saferkid	AAID	1 (bugfender.com)

⋆: Number of domains limited to 2 to fit display; AAID refers to Android Advertising ID;

We use the word "domain" to refer to second-level domains.

Chapter 5

Concluding Remarks

In this chapter, we deliver our recommendations to parental control solution providers to enhance the security of their products. We then discuss potential aspects of our work which could be the subject of future research. Finally, we present our conclusion to this thesis.

5.1 Recommendations

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

Addressing vulnerabilities. Because of the sensitivity of the information processed by the parental control solutions, companies should conduct regular security audits; the issues we listed in Sec. 3.1 can serve as a starting point. Moreover, they should have an identified process to address vulnerabilities such as responsible disclosure and bug bounty programs. During our analysis, among the solutions tested, only Kaspersky and Bitdefender were found to participate in such programs.

Enforcing best practices. Parental control companies should rely on publicly available guidelines and best practices, including proper API endpoint authentication and web security standards [28, 29]. We also strongly encourage companies to adopt a strong password policy in their products. The use of default, weak and stolen credentials has been exploited

in many known data breaches [72]. In the case of network devices, manufacturers should employ a secure firmware update architecture (see e.g., IETF [48]). Adopting known best practices is critical due to the especially vulnerable user base of these products.

Monitoring account activities. Parental control solutions 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 solutions should limit the collection, storage, and transmission of the children's data to what is strictly necessary. For instance, they should not store PII that is not required for the solution's functionality. The parental control solutions should allow the parent to selectively opt-out of the data collection in certain features.

Securing communication. Transmissions of PII should happen exclusively over secure communication channels. The solution should also utilize MITM mitigation techniques such as host white-listing, certificate pinning, and HSTS [36].

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

5.2 Future Work

In this thesis, we designed and applied a novel multi-platform security and privacy analysis framework for parental control solutions. By investigating a representative sample of parental control solutions through the prism of eight prevalent security threats (described in Sec. 3.1), we showed that this type of solutions is prone to security issues. This study can serve as a basis for further work on parental control solutions. In particular, the following aspects can be the subject of future research:

- In our study, we have prioritized the most popular platforms, namely Windows for personal computer OS, Android for mobile OS, Chrome for web browsers. It may be interesting to study the potential differences that would arise from the analysis of other types of solutions, such as iOS apps or Mozilla Firefox and Opera web extensions. Another noteworthy research scope concerns the OS-based parental control features mentioned in Sec. 2.1.3.

- The framework could also be improved, in particular by refining certain aspects of the analysis. In absence of former comprehensive analysis of parental control solutions, we drew inspiration from past work (see Sec. 2.2) to build our threat model. Further reverse engineering work on parental control solutions may uncover additional distinctive vulnerabilities in parental control solutions. The design and the application of a precise methodology for reverse engineering analysis of this type of solutions could be an interesting area for improvement.

- The analysis can also benefit from further automation, opening the door to a large extension of the analysis dataset. This especially concerns the parental control solutions grouped in software stores (mobile and web browser) that can easily be identified and extracted. The mobile app analysis part of our framework is already assisted by partial automation for the detection of PII leakage, and communication with known third parties. This effort can be extended to the other aspects of our framework.

5.3 Conclusion

In this thesis, we carried out a security and privacy analysis of parental control solutions. Those solutions are used by parents to help them protect their children from online risks. Some of these solutions have made news in recent years following significant data breaches. Nevertheless, only a selected number of academic research

(see Sec. 2.2) has been conducted on this topic. To contribute in this area of research, we designed an experimental framework for systematic analysis of security and privacy issues in parental control solutions and set up a cross-platform comprehensive analysis of this type of solutions. The results of our study showed systematic problems in the design and deployment of 49 out of 54 tested solutions and revealed a total of 170 security and privacy issues. Several identified flaws had the potential to directly affect the real-world safety of children in addition to the leakage of private information. These solutions are viewed as an essential instrument to provide children a safer online experience by many parents, yet we highlight that the majority of examined solutions broadly fail to adequately preserve the security of its users. We advocate that these solutions should be subjected to more rigorous and systematic evaluation and more stringent regulations.

Bibliography

[1] 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 *ACM CSS*, 2015.

[2] 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.

[3] O. Alrawi, C. Zuo, R. Duan, R. P. Kasturi, Z. Lin, and B. Saltaformaggio. The betrayal at cloud city: An empirical analysis of cloud-based mobile backends. In {*USENIX*} *Security 19*, 2019.

[4] 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)

[5] 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)

[6] 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)

[7] 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)

[8] Anton Skshidlevsky. Linux deploy. `https://github.com/meefik/linuxdeploy/`.

[9] M. Backes, S. Bugiel, and E. Derr. Reliable third-party library detection in android and its security applications. In *ACM SIGSAC CCS*, 2016.

[10] A. Beggs and A. Kapravelos. Wild extensions: Discovering and analyzing unlisted chrome extensions. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.* Springer, 2019.

[11] A. Bellissimo, J. Burgess, and K. Fu. Secure software updates: Disappointments and new challenges. In *USENIX HotSec*, 2006.

[12] broadband genie. Wi-Fi router security knowledge gap putting devices and private data at risk in UK homes. https://www.broadbandgenie.co.uk/blog/20180409-wifi-router-security-survey.

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

[14] Q. Chen and A. Kapravelos. Mystique: Uncovering information leakage from browser extensions. In *ACM SIGSAC CCS*, 2018.

[15] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy*, 2010.

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

[17] Chromium Blog. Project strobe. Online article. https://blog.chromium.org/2019/07/project-strobe-updates.html.

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

[19] 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.

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

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

[22] 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.

[23] 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/.

[24] Z. Durumeric, E. Wustrow, and J. A. Halderman. Zmap: Fast internet-wide scanning and its security applications. In {*USENIX*} *Security*, 2013.

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

[26] 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.

[27] Á. Feal, P. Calciati, N. Vallina-Rodriguez, C. Troncoso, and A. Gorla. Angel or devil? A privacy study of mobile parental control apps. In *PETS*, 2020.

[28] O. Foundation. HTML5 security cheat sheet. https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#local-storage.

[29] O. Foundation. REST security cheat sheet. https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html.

[30] Fowler, Geoffrey. Perspective | i found your data. it's for sale. Online article. https://washingtonpost.com/technology/2019/07/18/i-found-your-data-its-sale.

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

[32] T. Gerbet, A. Kumar, and C. Lauradoux. A privacy analysis of google and yandex safe browsing. In *DSN*. IEEE, 2016.

[33] Goodin, Dan. Google chrome extensions with 500,000 downloads found to be malicious. Online article. https://arstechnica.com/information-technology/2018/01/500000-chrome-users-fall-prey-to-malicious-extensions-in-google-web-store/.

[34] Google. Multiple Extensions are Compromised in a Browser Hijacking Scam. https://support.google.com/chrome/thread/46798301.

[35] M. Green. How safe is apple's safe browsing? Online article. https://blog.cryptographyengineering.com/2019/10/13/dear-apple-safe-browsing-might-not-be-that-safe.

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

[37] 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/.

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

[39] A. Le, J. Varmarken, S. Langhoff, A. Shuba, M. Gjoka, and A. Markopoulou. Antmonitor: A system for monitoring from mobile devices. In *ACM SIGCOMM C2B(I)D*, 2015.

[40] G. Leurent and T. Peyrin. From collisions to chosen-prefix collisions application to full SHA-1. In *EUROCRYPT*. Springer, 2019.

[41] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson. You've got vulnerability: Exploring effective vulnerability notifications. In {*USENIX*} *Security*, 2016.

[42] X. Li, C. Wu, S. Ji, Q. Gu, and R. Beyah. HSTS measurement and an enhanced stripping attack against HTTPS. In *SecureComm*. Springer, 2017.

[43] 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 *NDSS*, 2019.

[44] 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.

[45] M. Marlinspike. SSLStrip attack. https://github.com/moxie0/sslstrip.

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

[47] MOBSF. Mobile-security-framework-mobsf. https://github.com/MobSF/Mobile-Security-Framework-MobSF.

[48] 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.

[49] Nmap.org. Nmap network mapper. https://nmap.org/.

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

[51] OpenWrt Project. OpenWrt. https://openwrt.org/.

[52] OWASP Foundation. Insecure Direct Object Reference. https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html.

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

[54] 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/.

[55] 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.

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

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

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

[59] A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson. Haystack: In situ mobile traffic analysis in user space. *arXiv preprint arXiv:1510.01419*, 2015.

[60] B. Reaves, J. Bowers, N. Scaife, A. Bates, A. Bhartiya, P. Traynor, and K. R. Butler. Mo (bile) money, mo (bile) problems: Analysis of branchless banking applications. *ACM TOPS*, 2017.

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

[62] 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. *PETS*, 2018.

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

[64] 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/.

[65] S. Shasha, M. Mahmoud, M. Mannan, and A. Youssef. Playing with danger: A taxonomy and evaluation of threats to smart toys. *IEEE Internet of Things Journal*, 2018.

[66] O. Starov and N. Nikiforakis. Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions. In *Proceedings of the 26th International Conference on World Wide Web.*, 2017.

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

[68] 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.

[69] Unicef. Children at increased risk of harm online during global COVID-19 pandemic. Press release (Apr. 14, 2020). https://www.unicef.org/press-releases/children-increased-risk-harm-online-during-global-covid-19-pandemic.

[70] Unicourt.com. Blacklists université toulouse 1. Case Number: 3:17-CV-04419, filed on Mar. 8, 2017. https://dsi.ut-capitole.fr/blacklists/index_en.php.

[71] 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.

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

[73] e. a. Weissbacher, Michael. Ex-ray: Detection of history-leaking browser extensions. In *Proceedings of the 33rd Annual Computer Security Applications Conference.*, 2017.

[74] Weissbacher, Michael. These chrome extensions spy on 8 million users. Online article. https://mweissbacher.com/2016/03/31/these-chrome-extensions-spy-on-8-million-users/.

[75] 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.

[76] 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.

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

[78] 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 *ACM CSCW*, 2017.

[79] 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/.

[80] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu. Homonit: Monitoring smart home apps from encrypted traffic. In *ACM SIGSAC CCS*, 2018.