BABEŞ-BOLYAI UNIVERSITY
CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION DISTRIBUTED SYSTEMS IN INTERNET

DISSERTATION THESIS

LAN Level Parental Control

Author:
Sergiu Breban

Supervisor:
Dr. Dan Cojocar

June 2018

UNIVERSITATEA BABEŞ-BOLYAI
CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA SISTEME DISTRIBUITE ÎN INTERNET

LUCRARE DE DISERTAŢIE

Control parental la nivelul rețelei locale

Absolvent:
Breban Sergiu

Conducător ştiinţific:
Dr. Cojocar Dan

Iunie 2018

BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

Abstract

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION DISTRIBUTED SYSTEMS IN INTERNET

Master's degree

LAN Level Parental Control

by Sergiu Breban

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Contents

List of Figures

Chapter 1

Introduction

1.1 Parental controls

1.1.1 Overview

Parental controls [28] are the names for a group of settings that gives control to the parent of the content the children can see online. It developed in the digital era as a means to allow parents to restrict the access of content to their children and may be included in digital television services, computer and video games and mobile devices. More than 90% of parents of 5-15s who use parental control software consider it useful [26]. The content may not be appropriate for their age and is aimed more at adult audiences. The characteristics of inappropriate content depends for each parent, and is also correlated with the child's age and maturity level and includes information and images that can upset the child, inaccurate information or information that can cause dangerous behavior. Such content can be classified in the following categories:

- pornographic material
- content containing swearing
- sites that encourage vandalism
- pictures, videos or games which shows images of violence
- gambling sites
- unmoderated chatrooms

It is very easy for the child to stumble upon unsuitable sites by accident on any Internet enabled device, like mobile phone or tablet and it can be difficult to monitor and filter the content [18].

Parental control solutions fall into four categories:

- content filters, which limit access to different types of inappropriate content
- usage control, which works by constraining the usage of certain devices by placing time-limits on usage or forbid some types of usage

- computer usage management tools, which enforces the use of certain software
- monitoring, which can track the activity when using the devices

The rising availability of the Internet increased the demand for methods of parental control that restrict content. Mobile phones offer the most convenient and constant method for content access, and teens ages 13 to 17 are going online frequently. A study by Pew Research Center found that 92% of teens report going online daily, 24% of which are using the Internet almost constantly, 56% going online several times a day and 12% reporting once a day use. Only 6% go online weekly and 2% less often [19].

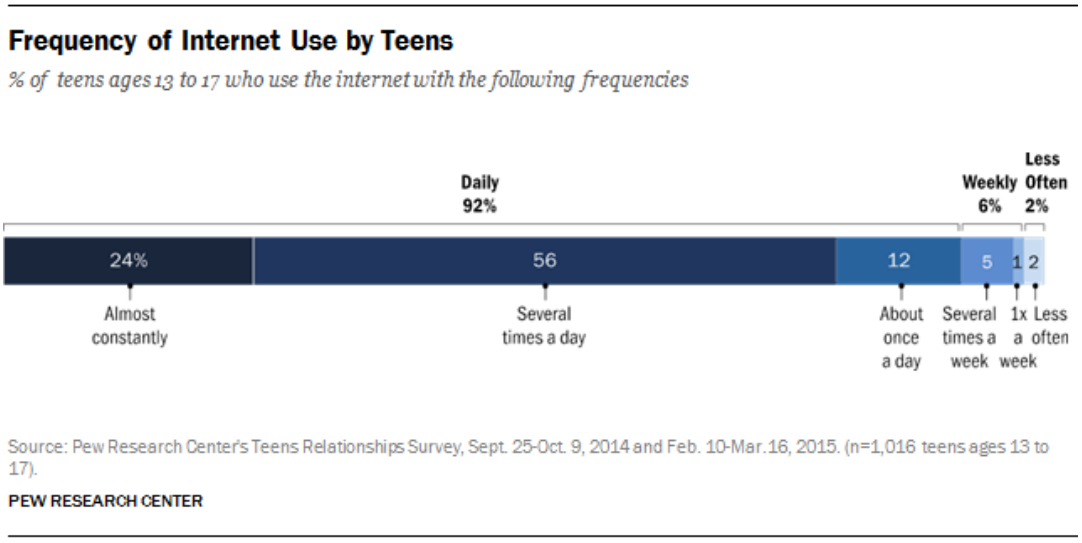


FIGURE 1.1: Frequency of Internet Use by Teens

The same study finds that nearly three-quarters own or have access to a smartphone and only 30% have a basic phone and 12% of teens, 13 to 17, have no cell phone of any type.

1.1.2 Techniques

There are two types of control techniques:

- behavioral control, which consists of controlling the amount of time and how much the child can consume;
- psychological control, which involves parents tying to influence children by affecting their emotional side by manipulating or insensitivity;

Adult control can be divided into three prototypes, each of which has influenced greatly the child-rearing practices [4]:

- permissive: the parent attempts to behave in a nonpunitive, acceptant and affirmative manner and consult with the child about policy decisions and gives explanations for rules

- authoritarian: the parent attempts to shape, control and evaluate the behavior of the child in accordance with a set standard of conduct, by valuing obedience as a virtue and favoring punitive, forceful measures to curb self-will

- authoritative: the parent attempts to direct the child's activities in a rational manner, by sharing the reasoning behind the policy and soliciting the child's objections when he refuses to conform; disciplined conformity and autonomous self-will are valued by the authoritative parent

Several techniques exists for creating parental controls to block certain websites. Parental control software can monitor API to observe applications such as web browsers or chat applications and to intervene based on certain criteria, such as time based criteria or as a match in a database of banned words. Other techniques that involve a proxy server are also used, in which the proxy server serves as an intermediary. The proxy server can use content based filters to intervene in the delivery of some content, but this method has a major disadvantage because it requires client configuration to use the service, which can be easily bypassed.

The difference between content filters and computer usage management methods is that the later is focused on empowering the parents to balance the computing environment for children. It does so by allowing parents to enforce the learning component into the computing time of children, where children can earn play time by working through educational content. This method is very powerful because it stimulates self-regulation in children, instead on relying solely on parental control, and we will use some ideas from this method to develop our control and regulation system.

Recently, some devices which are used for network based parental control have emerged. These devices use different methods to block inappropriate content, such as packet filtering, DNS Response Policy Zone (RPZ) and Deep packet inspection (DPI) [45], and work as a firewall router. Some commercial and governmental communication networks use these methods also, but these type of devices were developed for home also, and are used to create a new home wireless network specifically designed for kids to connect to. We developed our system using the same approach, by creating a custom wireless network for different type of users, different age level children and parents, and by using the packet filtering and DNS techniques to manage content filters.

1.1.3 Content filters

The increased use of mobile devices has created a demand for parental controls for these devices. The first carrier which offered age-appropriate content filters was Verizon, in 2007. With the release of iPhone OS 3.0 in 2009, Apple introduced a mechanism to create age brackets for users, to block unwanted applications from being downloaded. Filtering options are also offered by

most Internet providers, to limit Internet browsing options and block unsuitable content. The software used to restrict or control the content a user is capable to access is commonly referred to as Internet filter or content filter.

The content access restrictions can be applied at different levels, from governments applying them nationwide, to ISP blocking it's clients and by a parent to a child's computer. The content filtering mechanism can be implemented at different levels also, either by software on the client computer or by using the network infrastructure such as proxy servers, DNS servers and firewalls, but a mix of these technologies provides the most complete content control coverage.

- Browser based filters are the most lightweight solution, and the filtering is done by using a third-party browser extension
- E-mail filters are commonly implemented using a statistical method, Bayesian filters [35], by acting on information contained in the mail body, headers or attachments
- Client side filters work by installing it as software on each target device
- Content-limited (or filtered) ISPs are service providers that offer access to only a set of Internet content, to implement government, regulatory or parental control over its subscribers
- Network-based filtering is done at the transport layer, by implementing a transparent proxy, redirecting user requests to it by a switch or a router, or at the application layer, by configuring the client to send requests directly to the proxy server [8].
- DNS-based filtering [39] is implemented at the DNS layer and works by preventing lookups for domains that do not fit within a set of rules, parental control or company rules
- Search-engine filters work by filtering out inappropriate search results, but if the client knows the URL for a specific content, he can access it without using a search engine; search providers offers child friendly versions of their engines, which filter content inappropriate for children from the search results

We implement a DNS-based filtering mechanism, to be able to easily filter out the content that is definitely harmful for a child and does not bring any value, while not being too restrictive and giving the children room to explore and find by themselves what values means and how the time is best spent on the device. The main reason for filtering is to protect the user from harmful content, but it is used also to block malware and other intrusive material, as adware, spam, computer viruses, spyware, which can be even more harmful to children. The first level of filtering that we try to do is to block ads by integrating with the open-source ad-blocking solution Pi-hole [30], which works as a DNS sinkhole to block advertisement and Internet trackers.

Filtering mechanisms are not always efficient, and are also subject to some criticism. The filtering errors are of two kinds, overblocking, when the filter is too zealous and mislabels content that should be acceptable, such as labeling health related information as being porn-related, because of the Scunthorpe problem [1], and under-blocking, when the filter is unable to update quickly to new information available on the Internet. Content filtering can be a powerful censorship tool and there is a lot of discussion around the morality and legality of this kind of methods at a certain level, mostly state and country level. But it can be harmful if used incorrectly even at the family level, because using too strict policies can influence children behavior, and not always for good.

1.2 A self regulation approach

An in-depth study conducted on 75 Android apps that have the main purpose of promoting teens and children mobile online safety found that the majority of them (89%) are supported by features of parental control, and only 11% favour self-regulation [46]. The study presented a framework for Teen Online Safety Strategies which describes the difference between parental control and teen-self regulation. The three main parental control strategies identified are:

- monitoring is the surveillance of online activities, such as text messages, call logs or web browser history; Some studies found that monitoring was associated with higher online risks for some, suggesting to use monitoring only after some kind of online problem occurred [12].
- restriction occurs by placing rules on online activities, as setting limits on screen time and content acceptable for viewing. This kind of methods have some positive impact, such as reducing cyberbullying, but can also have negative effects, by causing children to take more risk-seeking behaviors [41].
- active mediation involves discussions regarding online activities between parents and children, and it reduces online risks without reducing the benefits of online engagement [12].

Self-regulation is the ability to control the emotions and behaviors by monitoring and evaluating oneself against given social standards. The analogous teen self-regulation strategies are:

- self-monitoring is a key component of self-regulation, and children must be aware of their own motivations and actions through self-observation [2].
- impulse control is the ability to inhibit short term desires in favor of the long term consequences caused by ones' actions, and losing control of this is the main reasons why self-regulation fails [3]

- risk-coping is a self-regulatory process that occurs after a stressful situation, by attempting to address the problem and the negative emotions caused. Actively coping with risky online situations help teens to feel less bothered about a risky event that occurred [9].

Usability was another issue in finding a good and efficient parental control mobile application. From the 89 application tested, 14 had configuration issues, some required users to have a Gmail account, other needed a VPN connection configured, some where showing annoying ads. Other apps were not meeting the goal of protecting the children from online risks, most of them were focusing on regulating web browsing and social media was one of the least prevalent activity monitored, but research suggest that most online risks, at least for adolescents, are encountered through the use of social media platforms. The features offered did not promote any values like trust, accountability, respect and transparency.

Most of the apps related to parental controls support mostly monitoring and restriction of mobile activities. Only some of the them support features like parental active mediation (<1%), teen risk-coping (4%), self monitoring (2%), or impulse control (<1%), but some of them added education as another safety strategy. We will try to have a different approach, by focusing more on the self-regulation approach while keeping the monitoring and restriction features to the minimum necessary, and try to add the education component also, to drive children to some learning resources and interactive quizzes before rewarding them with some device time.

We try to design our solution by following the practices suggested in the study from [46] and to focus on usability, social media control and to implement features that match the self regulation techniques. To be able to have more granular control and to implement specific features for each age, we create 4 age brackets, 0-5, 6-10, 11-13, and 14+. We try to design the application to be used mostly as a tool of communication between the children and the parents, not just a tool for enforcing the parents rules, by encouraging discussion about the rules being set. The process of establishing the rules and starting the parental control system should be done also as a collaboration between parents and children. The current solutions are fairly simplistic: as new functionality becomes available, new apps are created to regulate and monitor the children activities. While these approach prevent the risks, it also has the potential of limiting some positive engagement. As we can see in Figure 1.2, the new framework proposed for developing mobile online safety application is founded on core family values and emphasizes parental active mediation and self-regulation. The benefit of this framework is that it is not technically tied directly to children mobile activities and can focus on supporting more important needs of parents and children. We propose prototypes to promote collaborative practices between parents and children that support risk-coping and active parental mediation, by implementing a system for the children to learn about online risks and encouraging collaborative efforts when establishing policies. Another unique opportunity for design is in the are of supporting self-regulatory processes in the absence of parents. Instead of simply giving an SOS feature to get help from adults, some other

ways to support the children can be found, so that they can come up with their solutions to online problems or to come to the aid of others who could benefit from their help.

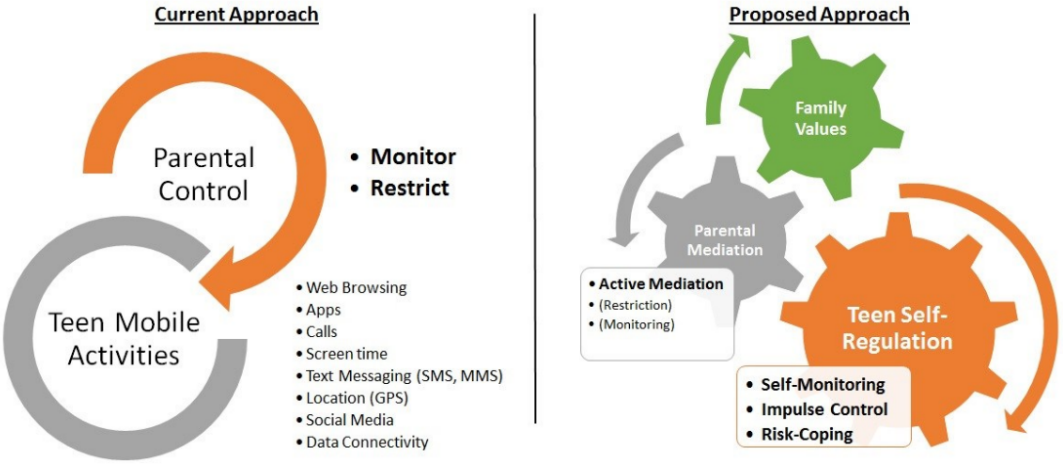


FIGURE 1.2: Current versus Proposed Approach for Online Safety Apps

Chapter 2

Content-control software and providers

Content control policies can be implemented at different levels. Some ISPs offer parental control options and even parental control software, other content control systems are integrated with the operating system, such as MacOS, which offers parental controls for some applications (Mail, Finder, iChat, Safari). The two major forms of content filtering technology are application gateway and packet inspection. The application gateway is called web-proxy for HTTP access and works by inspecting the request and the returned page using some rules and deciding if it should return the response. The packet inspection technique does not interfere with the connection, but inspect the data as it goes past and may decide at a later point to disconnect the client, by injecting a TCP-Reset or similar faked packet [10]. A combination of these two techniques is very popular because it allows more detailed filtering and can significantly reduce the cost of the system.

We will present next some parental control application and their features, on which we will try to build our system by taking a more self-regulatory approach.

2.1 Net Nanny

Net Nanny provides a content-control software as a way to monitor and control children's computer activity. The main features include blocking and filtering Internet content, place time limits on use and block PC games. Websites are blocked by content rather than URL, preventing children from accessing blocked websites through proxy websites. It is available on desktop platforms, Windows and Mac, and also on mobile platforms, Android and iOS, but the features and usability is not consistent on all platform, with some key features lacking on the mobile. It uses a dynamic filter that scans and analyses each web site to determine if it is appropriate for a child, based on a unique customization done by the parent. All the initial configuration is done online and it enforces the rules via a local client on each device it is installed. For each client, you can select from 4 profiles: Child, Pre-Teen, Teen and Adult, which can be further customized [24].

2.1.1 Content Filtering

Content filtering is the most used feature in all parental control systems and Net Nanny filters by analyzing the content for each web page in real time. Each site is matched against 17 objectionable categories, and each child profile have different level of blocking. Some categories are not completely blocked for some profiles, but the parents get a notification when the child visit sites in these categories. Parents can also create custom rules, to temporary allow access for some devices, and white-lists and blacklists for each child, to always allow or block certain websites. We follow the same approach by creating two classes of users for domain filtering and use a custom blocking mechanism to block certain domains for children, while allowing them for parent users [15].

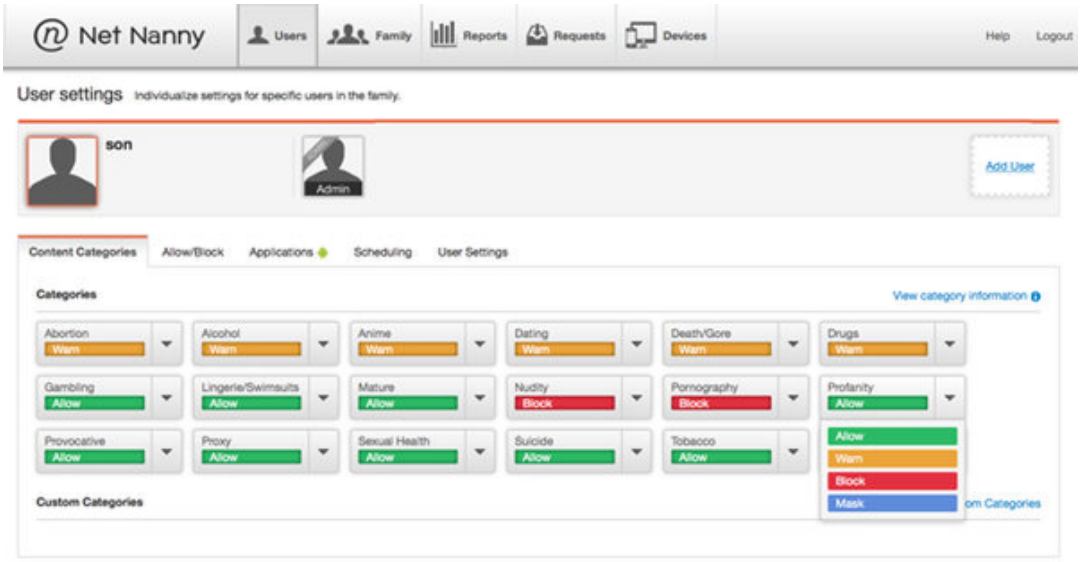


FIGURE 2.1: You can block content categories with Net Nanny's Web filtering tools.

2.1.2 Internet Time Scheduling

Children's Internet use can be controlled in two ways using Net Nanny, by creating weekly schedules in half-hour increments, and you can also create weekly allowance duration in one-hour increments. This system does not depend on the system clock, so it cannot be bypassed. But it would be more useful to have more granularity when setting time limits, that's why we support 15-minute increments allowance periods for each day.

2.1.3 Email notifications

Net Nanny have two types of email notifications enabled. The first type of notification is sent when a child request a blocking exception for a specific

domain, and the other one is in the form of a weekly summary. You can configure to get notification for multiple types of events, when a child hits a blocked site, continues after a warning or request a change in the blocking status. It would be more useful to have multiple options of getting notified about certain events occurring in the system, that's why we introduced mobile notifications into our system.

2.1.4 Detailed Reporting

You can use the built-in online console to view all the activity reports. An example report can be seen in Figure 2.2. You can view the activity by day, week and month, but not older than 30 days. It shows the blocked content in a pie chart, with some more information on mouse over. The report goes as deep as to show the page title, the time stamp and the user for each URL visited on a specific device. We do not include any type of detailed reporting, because we thing that it would be a privacy issue for the child, since we try to implement a self-regulation system, but we have only a small query log feature, to be able to see the most visited domains for each user category, in order to be able observe over-using behavior for some platforms and encourage discussion between parents and children. All the discussion should be done in the family and the child should be warned about visiting certain domains, but not by checking every move he makes online. That's why we don't include any location related features, and Net Nanny does not support location either, but some other parental control systems do.

2.1.5 Mobile Support

The mobile support for Net Nanny is similar to the desktop experience, with some limitations. For the system to work, all the browsing should be done through the proprietary browser offered, which can we see in Figure 2.3. On the iPhone the features are even more limited, because it does not interact with other apps and services at all. Since we started developing for mobile first and we try to keep the application features decoupled from the client device and system, we should not suffer of these limitations and we provide a seamless experience on both mobile platform, Android and iOS [21].

2.2 Qustodio

Qustodio [5] is a parental control tool that runs on any device, from PC to Android and iOS and even on Kindle devices. It has a lot of features, from web content filtering, app blocking and detailed activity logs. The configuration and monitoring is handled either through an online dashboard or a mobile control application. For configuration, you need to install a local client on every device and assign a child's profile. The Windows desktop client has even an option to hide the Qustodio install, but we do not support this kind of approaches, and that's way we choose to not hide the parental

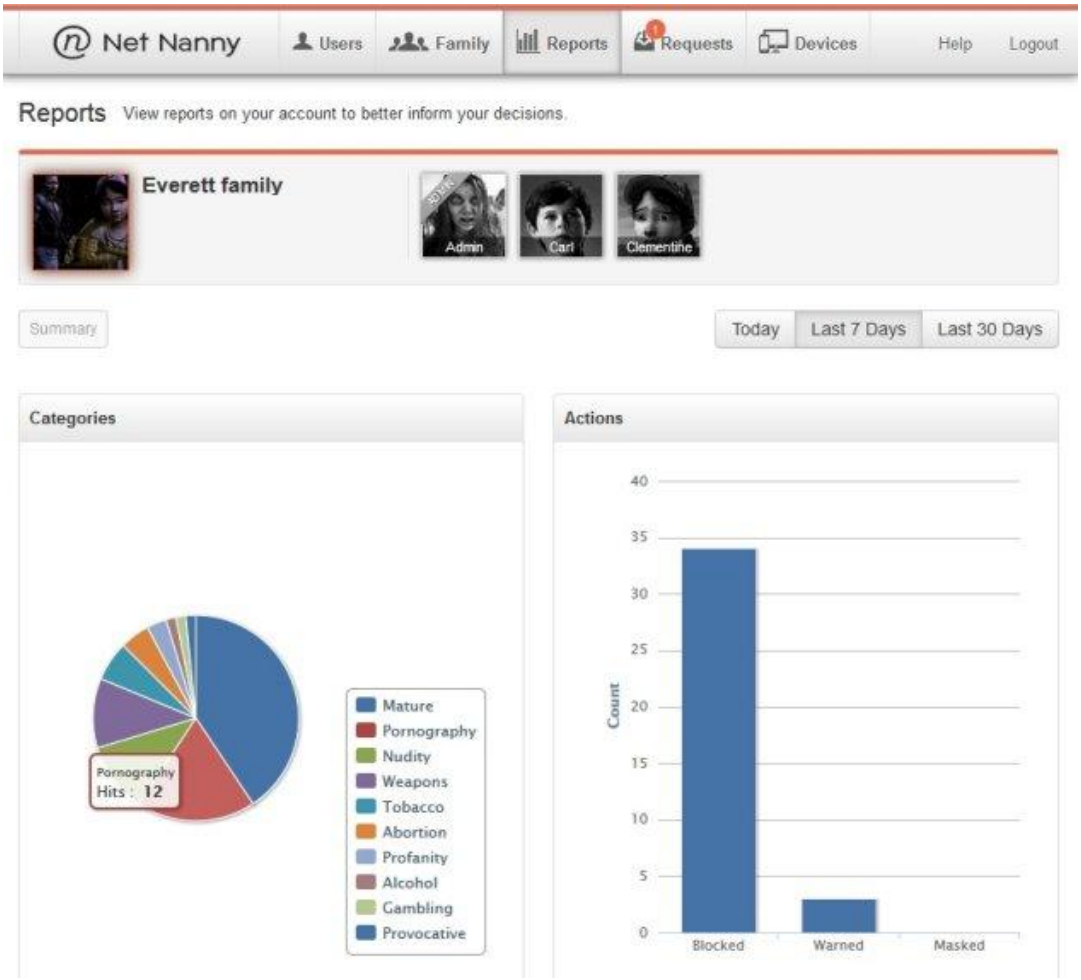


FIGURE 2.2: Net Nanny Reports

control app in any way, because there's a fine line between spying and parenting and when trying to follow a self-regulatory approach, conversation and transparency are more important. The process of registering the mobile app involves assigning an existing or a new child profile and a name for the device, and specifying whether it is a parent or a child device. We use the same registration process for a device and we also have 2 types of users, a parent and a child, each class having access to different features [22].

2.2.1 In-Depth Reports

You can use the online dashboard to get up to date reports, or you can get an email with the daily activity summary for each child. The usage overview for search, web, social, app and device is shown in an interactive chart, with information specific to each category, such as interactions on Facebook and visited URLs. There is also support for creating rules, such as web browsing rules, application rules and time-usage limits.

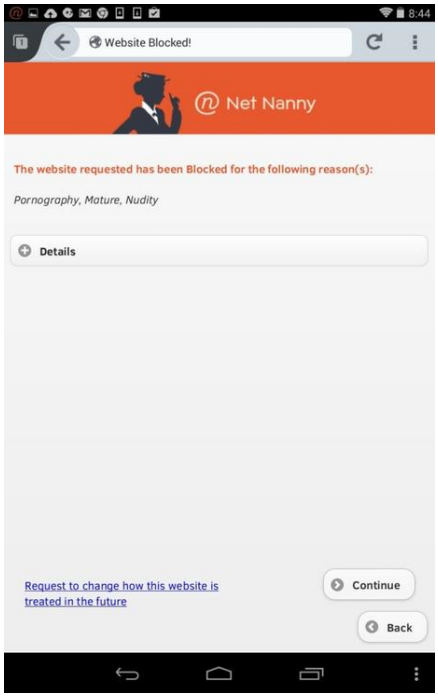


FIGURE 2.3: Net Nanny Mobile Browser

2.2.2 Web Filtering

The default configuration blocks all access for websites from ten undesirable categories, among them Drugs, Gambling, Pornography and Violence, and other 19 categories are available for more fine tuning. You can also configure an automatic notification when blocking a site. The content filtering is not dependent on the browser used and uses real-time analysis to supplement category database. It can also block HTTPS content, so it cannot be bypassed using an anonymizing proxy, but it does not have a feature to request temporary access for some blocked sites.

2.2.3 Time Usage Limits

You can control the usage by defining a weekly schedule in one-hour increments per device or by setting a daily maximum for each day. The time tracking is done cross device, so you can make sure that a child does not exceed its daily limits. The system can also block the device, not just control its Internet access. Access to any mobile application can be blocked, with some limitations on iOS. Time limits can be controlled at the application level, so you can limit the amount of time the child spends on certain social media apps during the school week.

2.2.4 Social Monitoring and Location Reporting

Social media monitoring on Qustodio is limited to Facebook. You can see the child's Facebook wall activity on the online dashboard, including posts,

FIGURE 2.4: Qustodio Reports

pictures and comments, as well as the identity of any friends involved in online chats, but it does not report the content of those chats, to maintain some degree of privacy. It also includes location-reporting features, which you can use to check the child's location as often as every five minutes.

2.2.5 Mobile Support

The mobile application on Android offers features similar to the online portal. You can view all the children associated with the account and also set up another user profile, but you can't remove a child or change what device is associated with what profile. It redirects to the web portal when trying to make these changes. On the Android device, it can monitor all calls and SMS messages and you can choose to record the content of each message, block calls or restrict specific contacts, but this does not apply to third-party messaging apps like WhatsApp. Another feature specific to Android is the Panic Button, which works by configuring up to four trusted contacts which get notified in case of emergency. The iOS version cannot block or monitor calls and texts, because iOS blocks most interactions between applications, but the other features are almost identical to the Android version, with some

more limitations related to location.[22] We chose to not implement any call and SMS related features because we think that it goes beyond the scope of parental control, and since our solution works only on the LAN level, we do not have any location tracking support.

2.3 Circle with Disney

Another system, which is more similar to ours, because it uses and external device connected to the local wireless network to implement parental control features is Circle with Disney [27]. It is very easy configurable, because you only have to install it on the network and then use the simple mobile app to configure time limits and content filtering, among other features. The initial configuration is simple, you only have to plug the device in and connect to the device's hotspot, using the password supplied by the app to link the local Wi-Fi to Circle. The technology it uses is called ARP (Address Resolution Protocol) spoofing [6]. The setup process is completed by configuring your own profile by choosing from five filter level: Pre-K, Kid, Teen, Adult and None. Next you have to identify the devices connected to the home network and choose a profile for them. You can leave some devices as unmanaged, and the others you can associated with the corresponding family member. You can customize each child's user profile with a photo and you can associate the devices owned to the user profile, but the system assumes that a device is used by only one child, so you cannot control properly shared devices [34].

2.3.1 ARP Spoofing

The Address Resolution Protocol (ARP) is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given network layer address, typically an IPv4 address and provides a critical function in the Internet protocol suite [29]. It is a request-response protocol which sends messages encapsulated by a link layer protocol and only communicates within the boundaries of a single network, never routing across inter-networking nodes. This property places ARP into the link layer of the Internet protocol suite [6]. ARP Spoofing, also called ARP cache poisoning [20], is a technique generally used by an attacker to send (spoofed) ARP messages onto a local are network. The aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead. It may allow the attacker to intercept data frames on the network, modify or stop all traffic, and is used to open other attacks, such as denial of service, man in the middle or session hijacking attacks [31]. This kind of attack can be used only on networks that use ARP and requires the attacker to have access to the local network segment [20].

When sending and Internet Protocol Datagram from one host to another in a local network, the destination IP address must be resolved to a MAC address, so and ARP request packet is broadcasted, and the machine with the IP

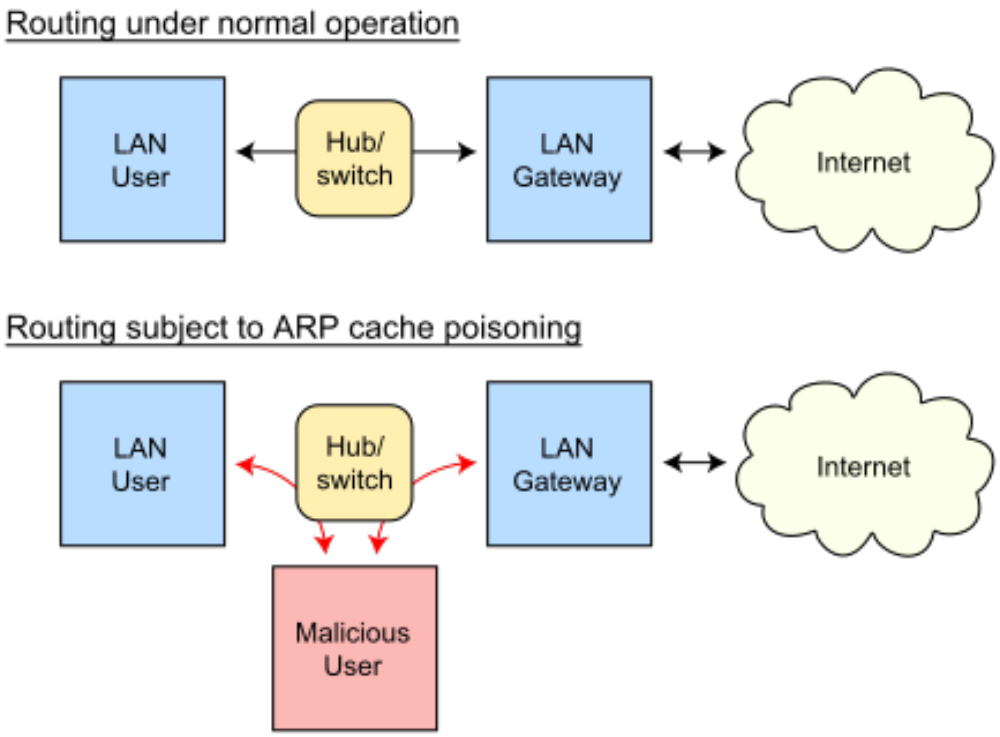


FIGURE 2.5: A successful ARP spoofing (poisoning) attack allows an attacker to alter routing on a network, effectively allowing for a man-in-the-middle attack

address from the request responds with an ARP reply that contains its MAC address. Since ARP is a stateless protocol, hosts will cache automatically the replies they receive and even ARP entries that have not expired yet will be overwritten when a new reply is received. The peer from which the packet originated have no method of authenticating with the host and this create the vulnerability that allows spoofing to occur. The techniques used in ARP spoofing are also used to implement redundancy of network services. Only two companies are known to date to have commercialized products centered around this strategy, and one of them is the parental control system, Disney Circle

2.3.2 Blocking Platforms

Each filter level comes with preconfigured platforms which you choose to block or unblock. Platforms refer to a list of Internet-aware apps. No platform is blocked at the Adult level, but for Teen level, HBO, Meerkat, Periscope, Reddit, Snapchat and Tumblr are blocked, for example, as we can see in the Figure 2.6 [43].

FIGURE 2.6: Circle With Disney Main Features

2.3.3 Content Filtering

There are 30 content categories which can be allowed or blocked, depending on profile. Every profile have some categories blocked by default, even the Adult profile has Dating, Explicit Content, Gambling, Mature and VPN & Proxies blocked. The Teen level blocks some more categories, while Kid level adds Social Media to the always-blocked list. For the Pre-K level, the only sites permitted are those from the Kids category. The changes to the filtering policies take effect immediately and since proxies are blocked by default, there is no way of getting around them. We chose to use the same approach and use an additional device on the network to implement the control policies, because it makes the experience device-independent.

2.3.4 Privacy and Safety

Circle offers also some privacy and safety settings, which are specific for each profile level. For example, Safe Search is forced in Google for Teen level, while the Kid level adds YouTube restrictions, and these restrictions are always on for Pre-K level. There is even and ad blocking feature, which works fine, without breaking the page layouts. Even though ad blocking hurts the revenue of websites relying on advertising, we think that kids should not be involved in the advertising business, so we decided to include the ad blocking feature, by relying on the Pi-Hole [36] system to block ads for children profile level.

2.3.5 Time Limits, Bedtime and Pause

Circle does not support weekly hour-by-hour Internet schedule the way other parental control systems do, but you can set a daily maximum for Internet access, which applies for all devices of a user. You can also set these daily limits for each platform or content categories. Internet access can be cut off for devices at a specified time, bed time, and will resume at the specified wakeup time. You cannot block access to any specific app, you can only cut the Internet access, since the device can control only the network. This can be a disadvantage, but is also more easy to configure such a system, because it may work without ever installing anything on the client device. Another feature that Circle offers is the pause function. You can choose to pause the Internet access for all the managed devices, only for one child's devices or just for one specific device.

Chapter 3

A self regulation approach

We started working on our system as a solution for parents who wanted a reliable parental control system, which tries to use the newest discoveries related to parental control impact on child development and is also easy to install and use. Most of the existing parental control systems are subscription based and work by creating a custom configuration for a specific family and enforcing the rules by using a client application for each target device. We wanted to design our system such that the user is in full control of the system configuration and all the data is kept locally, at the expense of the system working only at the LAN level. There is no point in getting the data outside the house, since its only use is inside the local network.

The system that we developed does not require any kind of client configuration, all the configuration needed is done on the central node, which in out case is a Raspberry Pi [32], which is an access point and a system control node, and a mobile application which the parents use to configure the parental control policies. We also offer a client application design for children devices, but that is only to extend the basic functionality and to also help with some initial configuration. So we basically moved the configuration part from the clients to the central node of the system. We tried to make the configuration as simple as possible, such that even a non-technical person can configure and start the system.

The only commercial level alternative control parental system that we found using the same approach is Circle With Disney, whose details we presented in the previous chapter. It uses an additional network device which you have to purchase and it work by connecting to the local wireless network and using a technique called ARP Spoofing [20]. Because we are building out system around the access point, instead of using an additional device, we have much more flexibility in the techniques we use for filtering and blocking at potentially a higher cost for the device. Any type of network device with a wireless card can be used to establish this kind of setup, but we used a Raspberry Pi because it is a cheap device, having almost the same price as a good router, and is quite powerful and flexible. The system diagram can be seen in Figure 3.1. The system was designed with handheld mobile clients in mind, like smartphones or tablets, but there is no limitation for other mobile computers, like laptops, since the same blocking rules and time usage limits would apply for that type of clients too.

Another difference between out approach and the Circle device is that

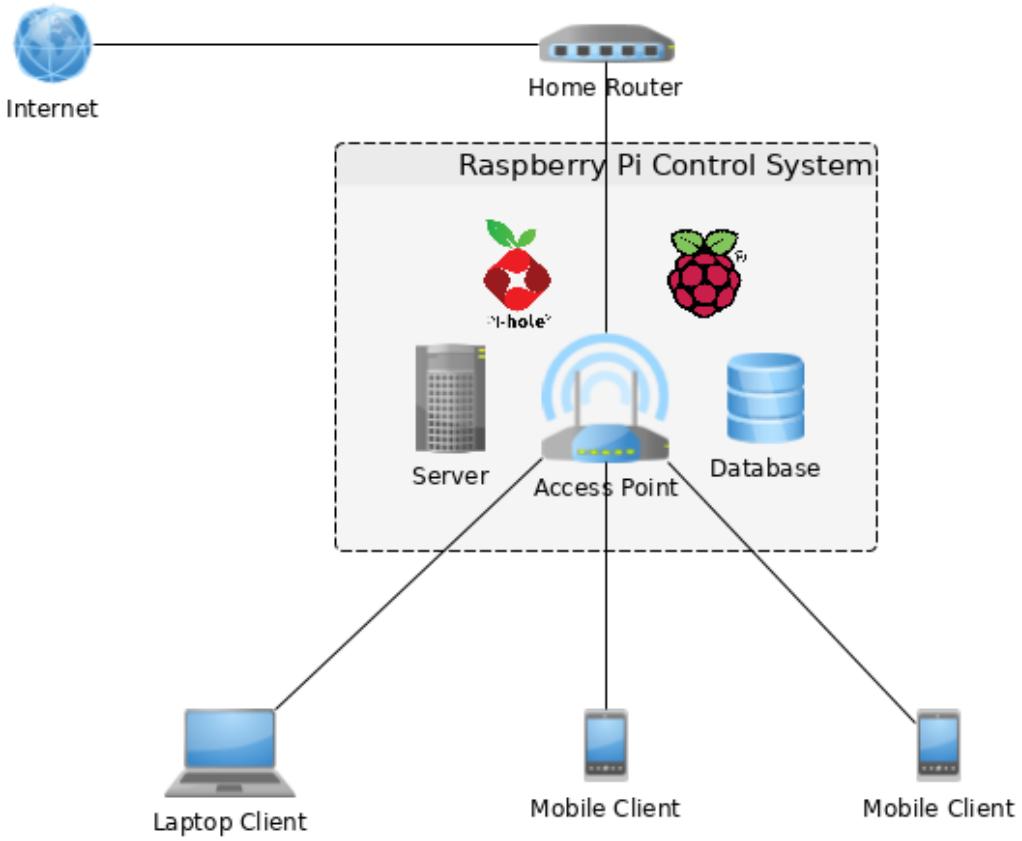


FIGURE 3.1: Raspberry Pi based parental control system diagram

the core of system is built around the Domain Name System, by using a local DNS server for filtering and blocking. For this test we are using the open source project Pi-hole, which uses the DNS forwarder and DHCP server Dnsmasq to create the blocking and filtering system, which serves the primary scope of blocking all ads. The framework used to develop the mobile application is Flutter [14], an open-source mobile application development SDK created by Google, to make the mobile control application available on both Android and iOS. We also used the Netfilter framework for more fine control on the network traffic on the access point. The tools and technologies used are as follows:

- Raspberry Pi 3 - as the access point and main controller
- Pi-hole - for blocking and filtering
- Dnsmasq - as DNS forwarder
- Netfilter - to control the traffic on the access point
- Go lang - the create a REST server for the mobile application

- Flutter - to implement the cross-platform mobile application

We will next present the principles behind each of these components of the system and details about how they all work together to make the parental control task easy for the parents and beneficial for children development.

3.1 Raspberry Pi

Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries [7]. All models feature a Broadcom system on chip (SoC) with an integrated ARM compatible central processing unit (CPU) and on-chip graphics processing unit (GPU). The model that we used for our setup is a Raspberry Pi 3 Model B, released in February 2016 with a 64 bit quad core processor and on-board WiFi, Bluetooth and USB capabilities. The presence of on-board WiFi card makes it suitable to run as an access point, the computing power is enough to run the setup that we need, the DNS forwarder and other tools. It also offers a rich software package, with multiple Linux-based operating system to choose from, the recommended version being Raspbian, a Debian-based system, which we use in our setup. The software community around the Raspberry Pi boards is also very active, which helps a lot in development and in troubleshooting any problems along the way. The Pi-hole system that we integrated into our system was also developed by the same community and created the opportunity for us to take it a step further and extend it by integrating the parental control features into it. We use the Raspberry Pi board as a wireless access point and also as a backend for the control system, since it is powerful enough to run the systems that we need. Out backend system has 4 main components: the Go server, the SQLite database, the Dnsmasq DNS forwarder and Pi-hole ad-blocking system, along with the quiz-taking system that we integrate, which is built using Ruby on Rails.

3.2 Pi-hole

Pi-hole is an Linux application which block advertisements and Internet trackers at the network-level, by acting as a DNS sinkhole, and also DHCP server, on a private network. A DNS sinkhole is the name gives to a DNS server that gives out false information to prevent the use of a certain domain name. It was designed for use on embedded devices with network capabilities, such as the Raspberry Pi. It has the ability to block traditional adverts on websites, but also adverts in other places, such as smart TVs and mobile operating systems. It works by serving as a DNS server for a private network and using a list of advert and tracking domains from predefined sources that the system uses to compare DNS queries to [36]. If a match is found in one of the block lists, Pi-hole will refuse to resolve the requested domain. This feature can

be used to block any domains, not the advertisement ones, by extending the block list.

We use this feature to block domains that are not considered useful for a child, such as social media platforms. You can also manually whitelist a domain, such that it is never blocked, and you can use the Pi-hole system to view reports about the DNS queries done by the clients. The main domain blocking feature works for all the clients, regardless of the age bracket, but we can establish certain domains for each age bracket that take priority in the visited domains reports. This approach is less restrictive for certain domains that are not bad, but does not bring much value for children either. We want the parent to know if the child spends to much time on these domains and to encourage discussion to find more valuable way of spending time online.

3.2.1 How it works

Pi-hole acts as a forwarding DNS server, so if it does not know where a domain is it forwards the query to another server. When you configure it, it knows where the ad-serving domains are, because they are all in the local list, along with all the explicitly blocked domains, so it does not have to forward those requests. But it does not know where the legitimate domains are, so those requests are forwarded to an upstream, recursive server. Those servers also don't know where it is only if they were asked to find it before, the only DNS servers that truly know where a domain is are the authoritative DNS servers. The flow of events that are triggered when Pi-hole receives a DNS query can be seen in the Figure 3.2 [37].

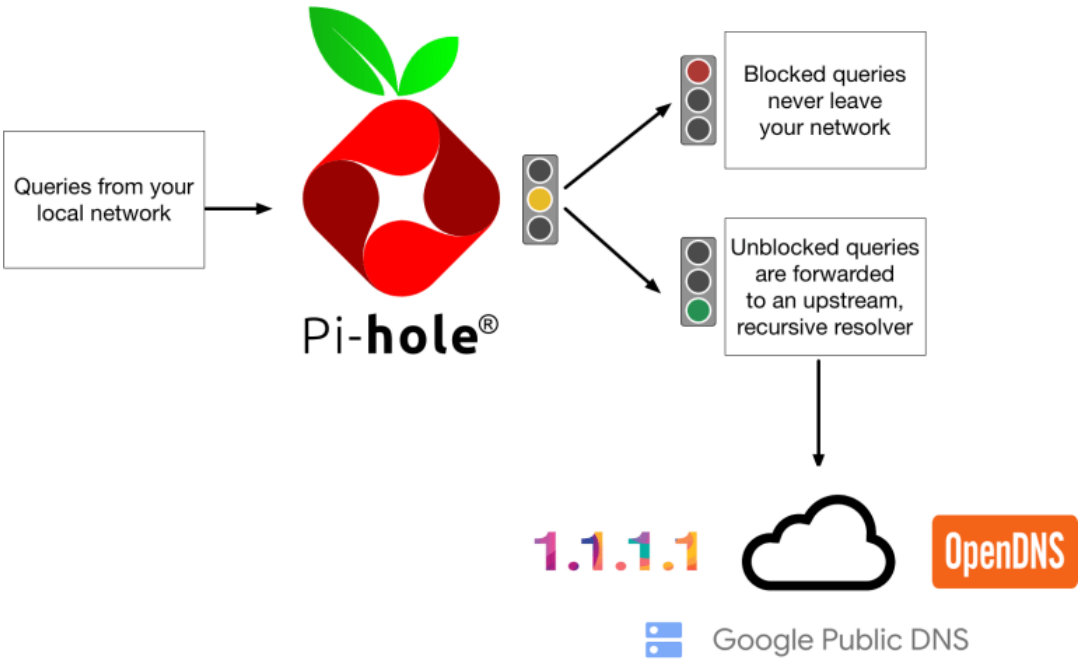


FIGURE 3.2: The flow of traffic through the Pi-hole system

The flow of traffic goes like this:

1. The client asks the Pi-hole who is pi-hole.net
2. Pi-hole will check the cache and reply if the entry is present in the cache
3. Pi-hole will check the blocking list and if the domain is blocked, it will replay accordingly
4. If neither 2 or 3 are true, Pi-hole forwards the request to the external upstream DNS server, that was configured on installation
5. After receiving the answer from the upstream server, Pi-hole will reply to the client
6. Pi-hole save the answer in the cache to be able to respond more quickly if the same domain is queried again

Pi-hole can also act as a network monitoring tool, since it logs all DNS queries sent to it, by default, so you can find what kind of traffic is going through your network. We integrated a component for visualizing this data into our system, to be able to quickly tell what domains are most visited by the children and what blocked domains they are trying to access. This integration was made easier by the fact that Pi-hole has a component which takes the raw data from the logs and pre-process it, storing it in a local database, to be able to serve quickly the types of queries needed, such as the top visited domains and the top blocked domains. We access this data by using a socket and some predefined commands. Pi-hole offers also a web interface to view these detailed reports. You can see the query types over time, forward destination over time, top domains, top blocked domains and other useful reports, as can be seen in the Figure 3.3.

Other benefits that Pi-hole brings to your local network are related to bandwidth and network speed. Because it works at the DNS level and prevents the ads and blocked domains from being downloaded. Since these ad images, videos and sounds are not being downloaded, the network will perform better. For the same reasons, it also reduces bandwidth, by preventing undesired assets from being downloaded. You can also extend the default block list to include sites that are known to serve malware or act as a phishing site, which brings another level of security, very important in the context of children using the Internet. The performance should not be a problem for the Pi-hole system, since even a not so powerful Raspberry Pi has been documented of handling up to 60 clients without problems, more than enough for a family network [38].

3.3 Dnsmasq

Dnsmasq is a lightweight and easy to configure DNS forwarder and DHCP server. It can be used to serve names for local machines which are not in the global DNS and is designed to offer DNS and DHCP solutions for small networks [17]. It has low resource requirements for system resources, so it is

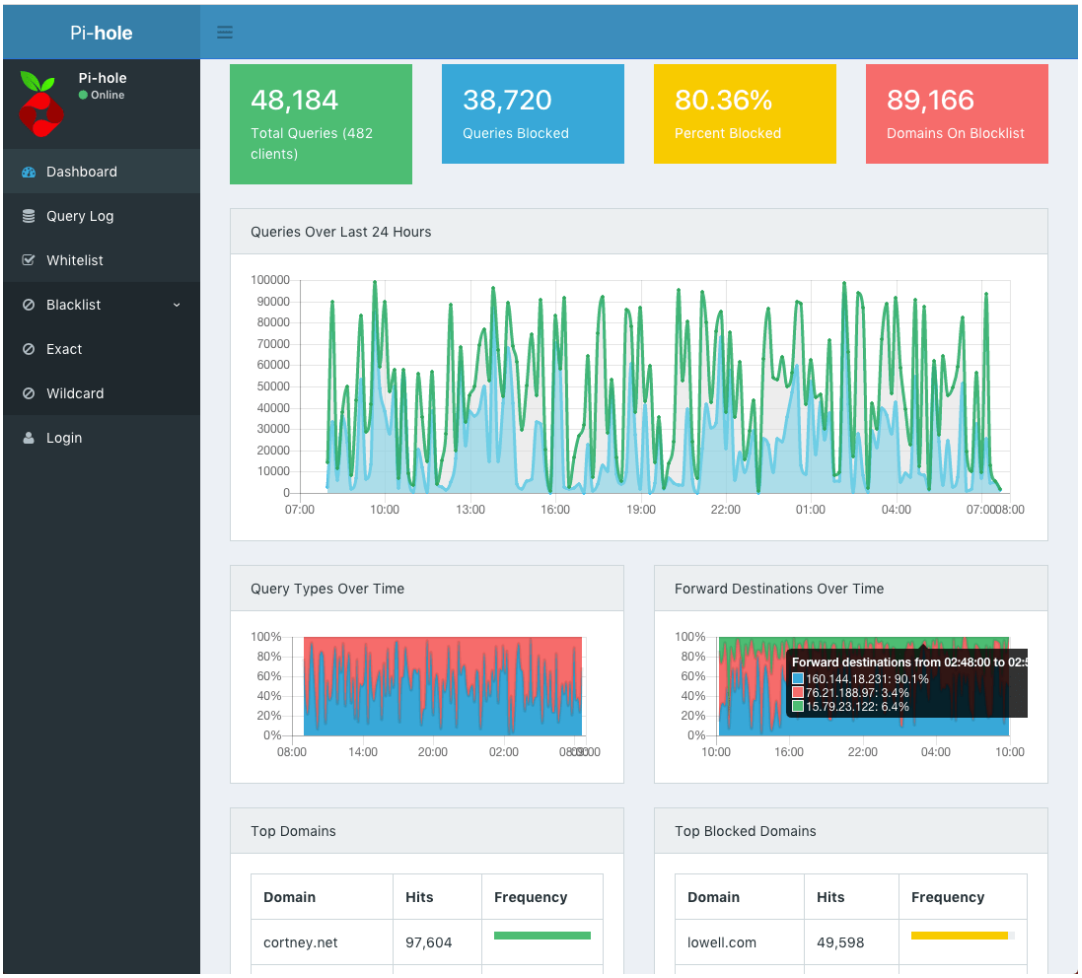


FIGURE 3.3: The Pi-hole admin dashboard

a good solution for our Raspberry Pi setup. It improves the performance and reduces the load on upstream DNS servers by caching DNS records locally.

Dnsmasq loads the content of /etc/hosts file to be able to resolve local names, and that allows some useful tricks, such as overriding some global domains, by adding a line with the format "0.0.0.0 annoyingsite.com". This prevents the domain annoyingsite.com from being resolved and that's the way Pi-hole uses it to block ads, by creating an entry for each known ad serving domain, and we also use that to block any other unwanted domains. Pi-hole keeps this Dnsmasq related configuration in a file named gravity.list, which is updated periodically to include the newest ad domains. It currently contains around 126000 of ad domains and the number is constantly growing. We can see below some of the line from this files, with 10 blocked ad serving domains.

```
1 192.168.0.103 zxypenguin.people-group.su
2 192.168.0.103 zy.zeroredirect1.com
3 192.168.0.103 zyban-store.shengen.ru
4 192.168.0.103 zyban.1.p2l.info
5 192.168.0.103 zyban.about-tabs.com
```

```
6  192.168.0.103  zybezradio.com
7  192.168.0.103  zyiztazhfprochain.review
8  192.168.0.103  zyjyyy.com
9  192.168.0.103  zyngatables.com
10 192.168.0.103  zyngawithfriends.com
```

The included DHCP server functionality supports static and dynamic DHCP leases, multiple networks and IP address ranges. We define a DHCP range for our network, along with the upstream DNS servers to use. The defined upstream DNS servers are the Google one and the configuration for this looks as follows:

```
1 server=8.8.8.8
2 server=8.8.4.4
3 interface=wlan0
4 dhcp-range=10.0.0.2,10.0.0.5,255.255.255.0,12h
```

Other Dnsmasq feature that is for use for us is that it can support different DNS server configuration for different DHCP settings. We are using this to establish the way the two classes of users defined in out system can access the network, since we don't want to apply the same blocking rules for the parents as for the children. Dnsmasq supports defining a different DNS resolver based on MAC address, and we use that to allow the parent's device to bypass the domain blocking rules that apply only to children devices, by configuring the system to use the Google resolver for that devices. That is easily done by adding the following rules to the Dnsmasq configuration:

```
1  ## This will go straight to Googles DNS Servers.
2  dhcp-option=tag:googlesdns1,6,8.8.8.8
3  dhcp-option=tag:googlesdns2,6,8.8.4.4
4
5  ## This will go straight to Opendns Servers.
6  dhcp-option=tag:opendns1,6,208.67.222.220
7  dhcp-option=tag:opendns2,6,208.67.222.222
8
9  ## Your Device that goes to Google DNS
10 #dhcp-host=dc:0b:34:cc:b0:ff,set:googlesdns1 #Nexus
11
12 ## Your Device that goes to OpenDNS
13 #dhcp-host=a8:b8:6e:41:0e:de,set:opendns1
14 #dhcp-host=f8:23:b2:ad:4d:f3,set:opendns1
```

We can configure different devices to work with different DNS servers, to add another level of control over the blocking features [11].

We are using the DHCP server functionality of Dnsmasq, to have control over what IP address gets every device and to easily identify the client devices by MAC and IP address, to be able to interpret the query logs and block access for specific devices. We establish a DHCP range for the clients of the access point, with a lease duration of 12h, but since all the logic works based on MAC address, the IP address is used only for display purposes. Because Dnsmasq is able to resolve local domains, we use that feature to block ad

domains and other domains unsafe for children. To enable access for the parent's devices to the blocked domains, we use a configuration with a different DNS server, the Google one, for the admin devices.

3.4 Netfilter

The Netfilter framework [25] allows various networking-related operations implemented in the form of customized handlers and is provided by any Linux operating system. It offers various packet filtering, network address translation and port translation features to provide the functionality required for directing packets through a network and also block packets from reaching certain locations within the network. Netfilter is represented by a set of hooks inside the Linux kernel which allows the registration of callback functions for specific kernel modules. These functions are applied as filtering and modification rules and are called for each packet that traverses the respective hook in the networking stack.

The most significant parts of the Netfilter hook system are the modules named ip_tables, ip6_tables and arp_tables. They provide a firewall system based on rules that can filter or transform packets. These rules are organized into tables which classify them according to the type of decision they are used to make. For example, there are rules dealing with network address translation and rules dealing with the decision to allow a packet to continue to its destination or not. The tables that iptables provides are [13]:

- filter table, used to make decisions whether a packet should be allow or not to reach its destination
- NAT table, used to implement network address translation rules, to determine if and how to modify the packet's source or destination address
- mangle table, used to alter the packet headers in different ways, such as adjusting the TTL (Time to Live) value
- raw table, with the purpose of marking packets in order to opt-out of connection tracking features built on top of the Netfilter framework
- security table, used to set internal SELinux security context marks on packets

Within each table, the rules are organized into separate chains, which corresponds to the Netfilter hooks that trigger them. The names of the built-in chains, with the corresponding hook, are [13]:

- PREROUTING, triggered by the NF_IP_PRE_ROUTING hook by any incoming traffic as soon as entering the network stack.
- INPUT, triggered by the NF_IP_LOCAL_IN hook after the incoming packet has been routed if destined for the local system

- FORWARD, triggered by the NF_IP_FORWARD hook after the incoming packet has been routed if it is to be forwarded to another host
- OUTPUT, triggered by the NF_IP_LOCAL_OUT hook by any outbound traffic created locally when it hits the network stack
- POSTROUTING, triggered by the NF_IP_POST_ROUTING hook after routing and before sending the traffic out on the wire

The complex flow of a packet through the Netfilter framework can be seen in the Figure 3.4, with all the layers and tables involved.

After configuring the Raspberry Pi to work as a wireless access point [40] by using the user space daemon hostapd [16], we have to enable IP forwarding for the wireless clients, such that they are able to access the Internet, not just the local network. This is done by changing the following setting in the /etc/sysctl.conf file:

```
1 net.ipv4.ip_forward=1
```

We need to also add a new rule for the outbound traffic on the eth0 interface, which provides the Internet connection to the Raspberry Pi. This is done by adding a POSTROUTING rule to mask the private IP address of a node with the external IP address of the gateway, using the MASQUERADE target:

```
1 sudo iptables -t nat -A POSTROUTING -o eth0 -j
    MASQUERADE
```

To give Internet access to the access point clients connected on the wlan0 interface we have to add the following filter FORWARD rule:

```
1 sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --
    state RELATED,ESTABLISHED -j ACCEPT
```

We also use iptables rules to allow or block all network access for specific devices and to enforce the time limits for them. To block access for a device we add an INPUT rule for a device with a specific MAC address to drop the packets, by running the command like:

```
1 sudo iptables -A INPUT -i wlan0 -m mac --mac-source 3C
    :83:75:D0:DA:C4 -j DROP
```

To allow back access for the blocked device, we run the command to remove the rule:

```
1 sudo iptables -D INPUT -i wlan0 -m mac --mac-source 3C
    :83:75:D0:DA:C4 -j DROP
```

These commands are also run by the Go back-end using a cron table, to enforce time limits for network access for devices.

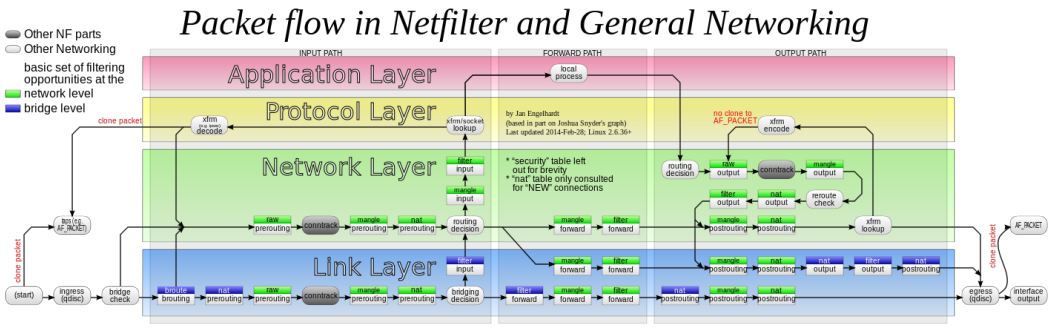


FIGURE 3.4: Flow of network packets through the Netfilter

3.5 Mobile application

In order to give the parents easy access on the network rules that the system support, we developed a mobile application. Using the application, a parent can easily establish the supported content control rules and view a short summary of usage for each device. The application has a simple client-server architecture, using a Go-lang back-end running on the Raspberry Pi and a cross platform mobile application, running on Android and iOS devices. The data needed to configure the system is minimal, so we used the relational database management system SQLite [42], since it is easily configurable and does not require powerful hardware to run on, which makes it suitable for the Raspberry Pi.

3.5.1 Database

Since not much data is needed for the system to work, the database diagram is fairly simple, as we can see in the Figure 3.5. The most important entries are the users, which represents a child, and the association between a user the the devices it owns. Each device is represented by the state it is in, blocked or not, and the time frame that if has access to the Internet. Each user corresponds to an age bracket, if it does not have the admin flag set. The age bracket is linked to a set of the blocked domains and this relation is used when showing the usage summary for each device, to establish a higher priority for the associated domains. This is used when the domains from the list are not blocked, in order to give the parent an idea about how much each child uses the domains that are considered not useful for its age and to initiate a discussion about this.

3.5.2 Server

The language of choice for the back-end system is Go [44]. Go is an open source programming language that makes it easy to build simple, reliable and efficient software. We chose this language because of its rich open source

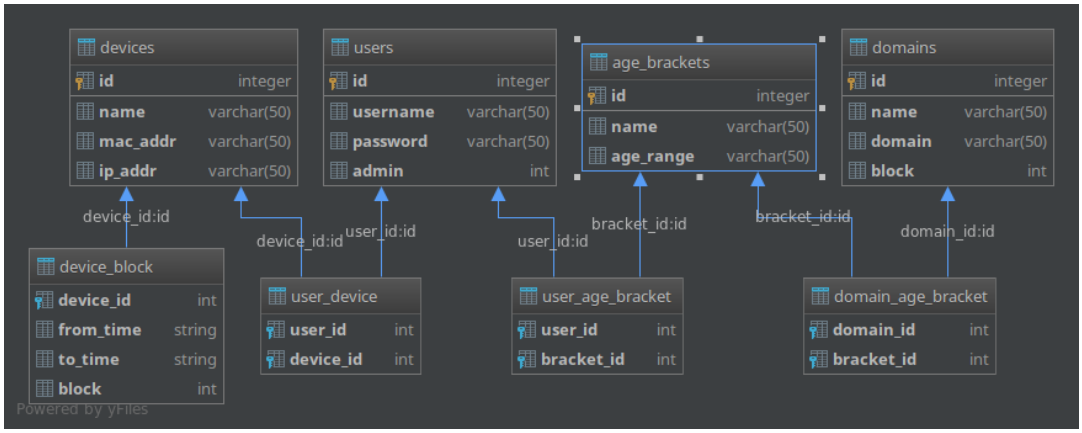


FIGURE 3.5: Database diagram

support, its powerful remote package management tools and the built-in concurrency primitives: light-weight processes (goroutines), channels and the select statement. This features provides a strong foundation for building a RESTful web service to run in a resource limited environment, like the Raspberry Pi. The rich open source support and libraries makes it easy to build such a system and quickly scale it.

At the root of our back-end system, we use the package gorrila/mux [23], which implements a request router and dispatcher for matching incoming requests to their respective handler. This package makes it easy to define multiple routes, represented by a pattern in the URL, and their corresponding handlers. We defined a Route struct to hold the necessary parameters, such as URL pattern and handler, and we initialize the router using a list of Routes. Some of the routes that we define are:

```
1  Route{
2      "Users",
3      "GET",
4      "/users/{userId}",
5      usersHandler,
6  },
7  Route{
8      "AddUser",
9      "POST",
10     "/add_user",
11     addUserHandler,
12 },
13 Route{
14     "Devices",
15     "GET",
16     "/devices/{userId}",
17     devicesHandler,
18 },
19 Route{
```

```
20    "RegisterDevice",
21    "POST",
22    "/register_device/{userId}",
23    registerDeviceHandler,
24  }
```

The router is created by a single command, and then the routes list is registered with the router, by adding the correspondig handler to each pattern:

```
1  router := mux.NewRouter()
2  for _, route := range routes {
3    router.Handle(route.Pattern,
4      jwtMiddleware.Handler(route.HandlerFunc)).
5      Methods(route.Method)
6  }
```

Mux supports the addition of middlewares to a Router, which are executed in the order they are added if a match is found, including its subrouters. Middlewares are (typically) small pieces of code which take one request, do something with it, and pass it down to another middleware or the final handler. Some common use cases for middleware are request logging, header manipulation. We use that to add a JSON Web Token (JWT) [33] middleware to the router, in order to be able to use an JWT as an access token to our API, such that only our mobile client can make requests. JWT is a compact, URL-safe means of representing claims to be transferred between two parties, and we use it to secure our API.

An example handler, which loads the domains defined in the system to support blocking and handles the /domains route, is:

```
1  func domainsHandler(w http.ResponseWriter, r *http.
     Request) {
2    db, err := sql.Open("sqlite3", "./clients.db")
3    checkError(err)
4    defer db.Close()
5
6    rows, err := db.Query("select␣d.id,␣d.name,␣d.domain,␣
       d.block␣from␣domains␣d")
7    checkError(err)
8    defer rows.Close()
9
10   var netDomains Domains
11   for rows.Next() {
12     var id int
13     var name string
14     var domain string
15     var block int
16
17     err = rows.Scan(&id, &name, &domain, &block)
18     checkError(err)
19     fmt.Println(id, name, domain)
```

```
20        domainElement := Domain{Id: id, Name: name, Domain:
              domain, Block: block}
21        netDomains = append(netDomains, domainElement)
22    }
23    err = rows.Err()
24    checkError(err)
25    json.NewEncoder(w).Encode(netDomains)
26    json.NewEncoder(os.Stdout).Encode(netDomains)
27 }
```

The back-end defines multiple router and handlers for them, in a way that corresponds to the routes we have on the clients side, so we will detail the routes and features in the next session.

Other packages used on the server side is JSON encoding package, because we use this open-standard file format to transfer data between the client and server, the SQLite integration package and the cron package, to implement the job runner that we need to establish the temporal Internet access rules.

3.5.3 Client details

We needed to make the parental control experience consistent on both major platforms, Android and iOS, and that's why we are using the mobile framework Flutter, a modern and cross-platform approach offered by Google. Flutter is open-source and is used for crafting high-quality native interfaces on iOS and Android. The development process is fast because of the rich set of fully-customizable widgets and the native performance is full because it incorporates all the critical platform differences.

The application UI is structured in a hierarchical manner. We have a high level menu with and entry for each main feature, like blocking and reporting. Under each main menu entry, we have multiple sub-menus corresponding to the related functionality. The navigation between different pages of the user interface is done using a router, which support query string parameter parsing and function handlers. We start by creating the router and other application configuration, and then navigating to the home page, different for a child or admin type user, but if the user is not logged in, it is presented with the log-in screen first. A list of routes, which we will detail in the following sections, can be seen in the listing below. These routes closely match the routes that we have defined on our REST web service, which is used to handle the client requests.

```
1 static String root = "/";
2 static String adminRoute = "admin";
3 static String userHomeRoute = "userHome";
4 static String registerDeviceRoute = "register_device";
5 static String usersRoute = "users";
6 static String addUserRoute = "add_user";
7 static String devicesRoute = "devices";
8 static String deleteDeviceRoute = "delete_device";
```

```
9  static String addDeviceRoute = "add_device";
10 static String getDeviceBlockRoute = "get_device_block";
11 static String setDeviceBlockRoute = "set_device_block";
12 static String domainsRoute = "domains";
13 static String addDomainRoute = "add_domain";
14 static String editDomainRoute = "edit_domain";
15 static String deleteDomainRoute = "delete_domain";
16 static String topDevicesRoute = "top_devices";
17 static String topDomainsRoute = "top_domains";
```

The navigation to a new route is done by calling navigateTo on the router object, with the context and the route pattern, including parameters, as arguments. We can see in the listing below how we navigate to the devices view for a user, by using the user ID as parameter to the route pattern:

```
1 router.navigateTo(context, "${Routes.devicesRoute}?
    userId=${user.id}");
```

For the user interface component, we build custom widgets by extending the StatefulWidget class to create lists, cards or forms that fits our model elements and hold state that we need for the application to work, such as domain or device blocking state. In the listing below we can see how we build a scrollable list view by loading all the devices for the a user, and the resulting UI component in Figure 3.6. The block icon on the right represent the state of the device, blocked or not, and that state changes on list item tap.

```
1  class DevicesWidget extends StatefulWidget {
2
3    final int userId;
4
5    DevicesWidget(this.userId);
6
7    @override
8    State createState() {
9      return new DevicesWidgetState(userId);
10   }
11 }
12
13 class DevicesWidgetState extends State<DevicesWidget> {
14   final _biggerFont = const TextStyle(fontSize: 18.0);
15   final int userId;
16
17   DevicesWidgetState(this.userId);
18
19   Widget _buildDevicesList() {
20     print("_buildDevicesList");
21
22     RestDataSource restDataSource = new RestDataSource()
         ;
```

```
23      final Future<Response> response = restDataSource.get
          ("${Routes.devicesRoute}/$userId");
24
25      return new FutureBuilder(
26        future: response,
27        builder: (BuildContext context, AsyncSnapshot<
            Response> snapshot) {
28          if (snapshot.data != null) {
29            try {
30              print("Devices response data: ${snapshot.
                  data.body}");
31              final responseJson = json.decode(snapshot.
                  data.body);
32              return new ListView.builder(
33                  padding: const EdgeInsets.all(16.0),
34                  itemBuilder: (context, index) {
35                    if (index < responseJson.length) {
36                      print("Device $index : ${
                          responseJson[index]}");
37                      if (responseJson[index] != null) {
38                        Device netDomain =
39                            Device.fromJson(responseJson[
                                index]);
40                        return _buildRow(netDomain);
41                      }
42                    }
43                  });
44            } catch (e) {
45              return new Text("Error loading: " + e.
                  toString());
46            }
47          } else {
48            return new CircularProgressIndicator();
49          }
50        },
51      );
52    }
53
54    Widget _buildRow(Device value) {
55      return new DeviceWidget(value, userId);
56    }
57
58    addDevice() {
59      var router = Config.getInstance().router;
60      router.navigateTo(context, "${Routes.addDeviceRoute
          }?userId=$userId");
61    }
```

```
62
63    @override
64    Widget build(BuildContext context) {
65      print("Build DevicesWidgetState");
66      return new Scaffold(
67        appBar: new AppBar(
68            title: new Text('Devices'),
69            backgroundColor: Colors.black87,
70            actions: [
71              new IconButton(
72                icon: new Icon(Icons.add),
73                onPressed: () => addDevice(),
74              ),
75            ]
76        ),
77        body: _buildDevicesList(),
78      );
79    }
80  }
```

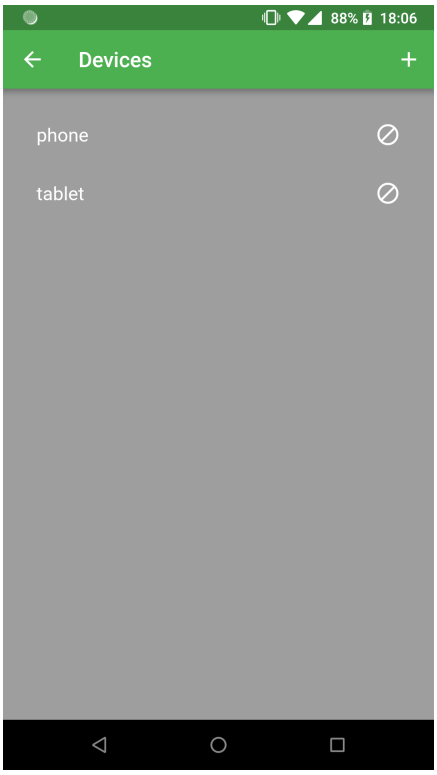
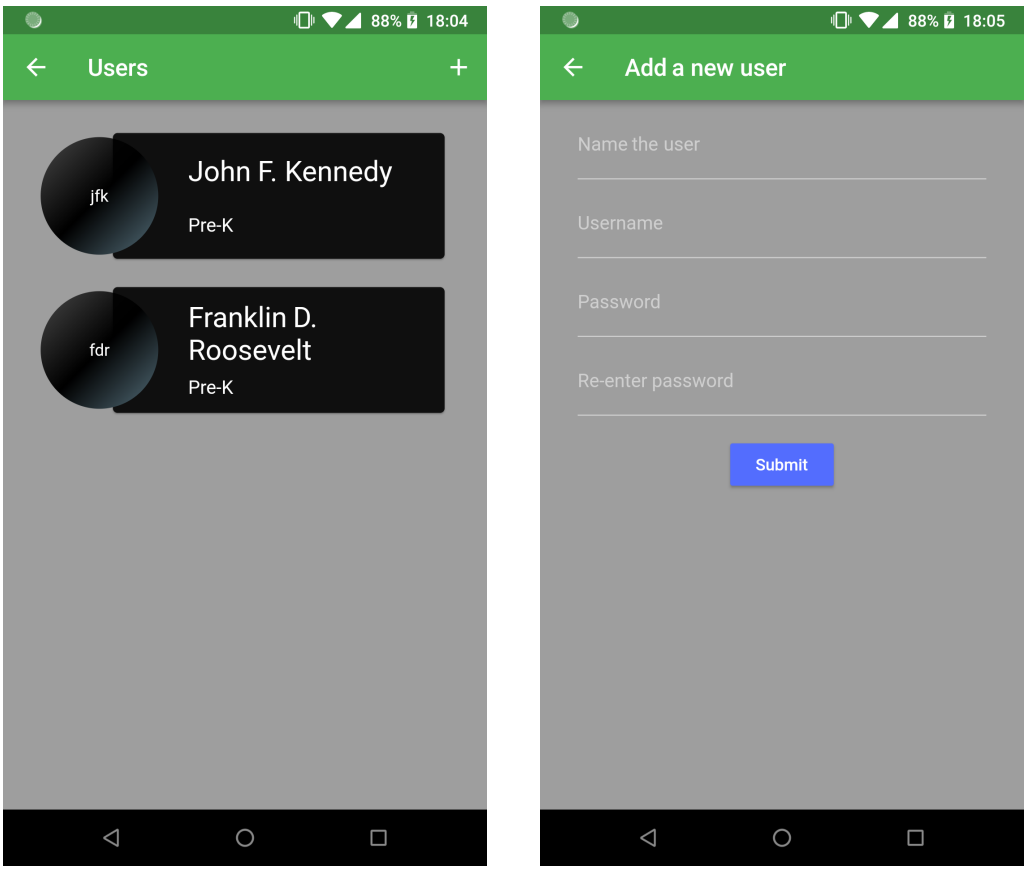


FIGURE 3.6: The devices view for a user



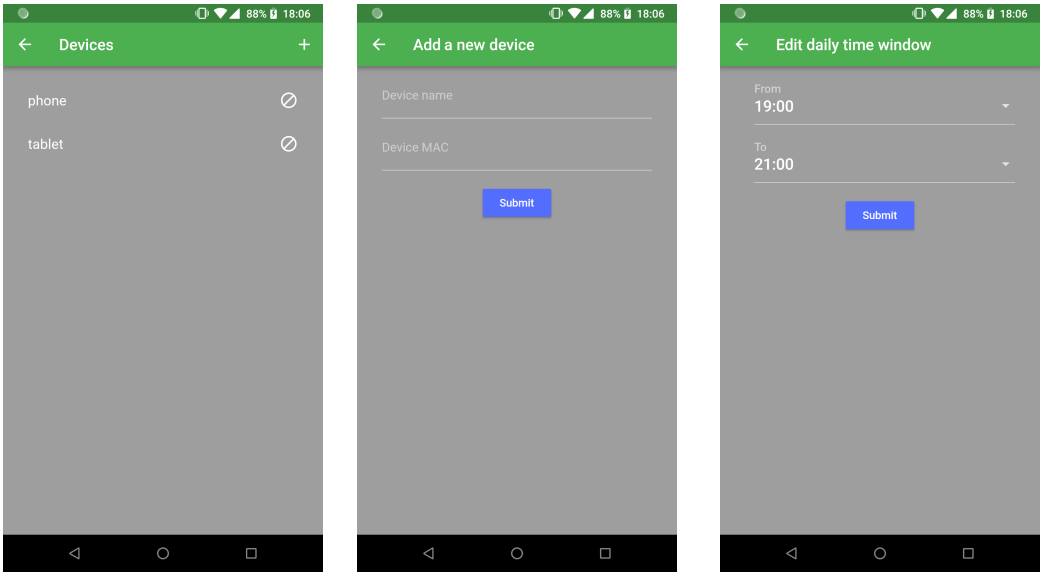
(A) The user view (B) The add user view

FIGURE 3.7: The user related views

3.5.4 Blocking and filtering

We provide two levels of filtering, for devices in particular and for domains in general, that are applied to all devices. The device level blocking is managed using the user UI. When accessing the user UI, we can see a list of users for which the current user is admin. Each user is shown in a card like view, with the profile picture, user name and the associated age bracket. We can also add a new user, by using the add menu from the top right corner. This menu gets us to the add page, to fill the user details. The user views can be seen in Figure 3.7.

For each user, we can view the list of associated devices by tapping on the user card. This gets us to a list of devices, which we can block or unblock, depending on the current state. A long press on a device shows a quick menu, from which we can access the time window setting component, where we can define the time interval in which the device has access to the Internet. That menu also allows the admin user to edit the details of a device or to delete it. We can also add a new device by using the top right menu item. All these view are shown in the Figure 3.8.



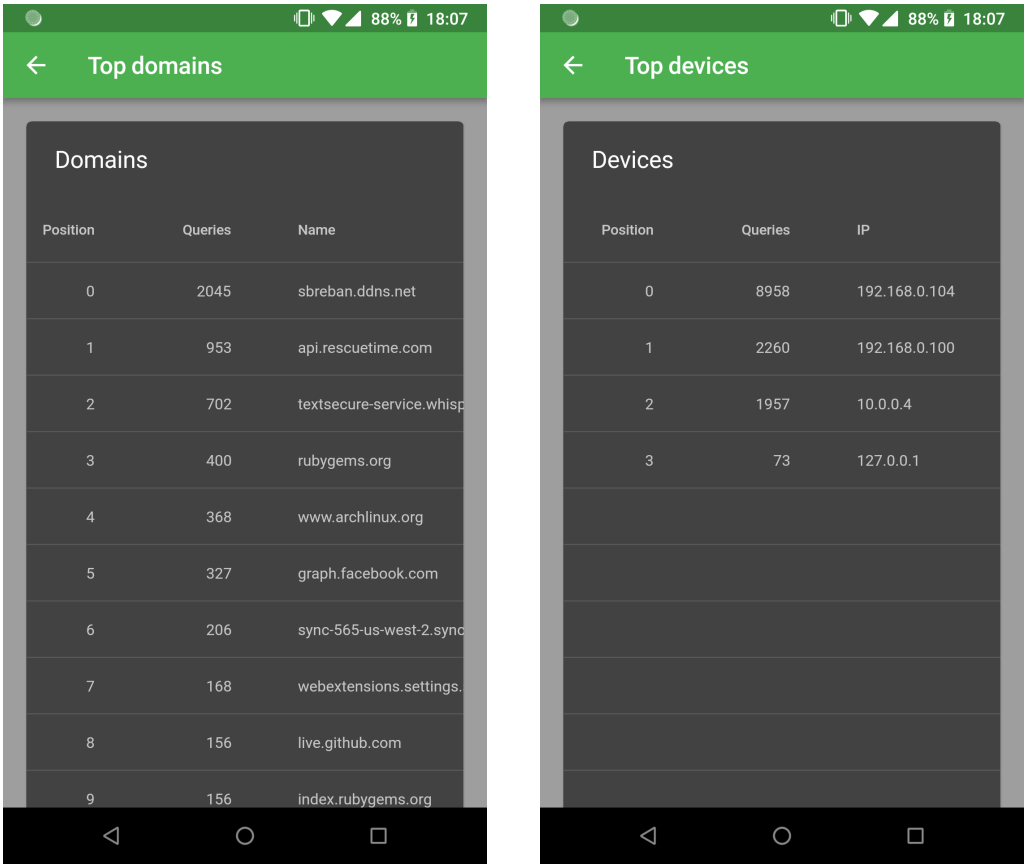
(A) The devices view (B) The add device view (C) Device time window

FIGURE 3.8: The user related views

3.5.5 Reporting

In contrast with the other parental control systems presented, we provide only limited reporting functionality, since we try to rely more on self-regulation than parental control. We try to encourage more communication between the parents and the children about Internet usage. The parent needs to understand some of the screen time usage of a child and he needs some data to get a starting point for the discussion. We try to provide this kind of data to the parent. This data is split into two categories: domains related query data and device related query data. The first category shows the top ten most visited domains, with the number of queries for each domains, while the second category show a top of devices with made the most requests to the DNS server. The related view tables can be seen in the Figure 3.9.

3.6 Self regulation component



(A) Top domains view (B) Top devices view

FIGURE 3.9: The reporting related views

Bibliography

[1] Tran Nhat Anh. "Scunthorpe Problem–Software Filters Misunderstand Words". In: (2015).

[2] Albert Bandura. "Social cognitive theory of self-regulation". In: *Organizational behavior and human decision processes* 50.2 (1991), pp. 248–287.

[3] Roy F Baumeister and Todd F Heatherton. "Self-regulation failure: An overview". In: *Psychological inquiry* 7.1 (1996), pp. 1–15.

[4] Diana Baumrind. "Effects of authoritative parental control on child behavior". In: *Child development* (1966), pp. 887–907.

[5] *Best Parental Control Software - Qustodio*. https://www.qustodio.com/en/. Accessed: 2018-02-28.

[6] Robert Braden. "RFC-1122: Requirements for internet hosts". In: *Request for Comments* (1989), pp. 356–363.

[7] Rory Cellan-Jones. *A 15 pound computer to inspire young programmers*. http://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html. Accessed: 2018-03-01.

[8] *Content Gateway explicit and transparent proxy deployments*. https://www.websense.com/content/support/library/deployctr/v78/dic_wcg_deploy_expl_trans.aspx. Accessed: 2018-05-30.

[9] Leen d'Haenens, Sofie Vandoninck, and Veronica Donoso. "How to cope and build online resilience?" In: (2013).

[10] Sarang Dharmapurikar et al. "Deep packet inspection using parallel bloom filters". In: *High performance interconnects, 2003. proceedings. 11th symposium on*. IEEE. 2003, pp. 44–51.

[11] *dnsmasq*. https://wiki.archlinux.org/index.php/dnsmasq. Accessed: 2018-06-09.

[12] Andrea Duerager and Sonia Livingstone. "How can parents support children's internet safety?" In: (2012).

[13] Justin Ellingwood. *A Deep Dive into Iptables and Netfilter Architecture*. https://www.digitalocean.com/community/tutorials/a-deep-dive-into-iptables-and-netfilter-architecture. Accessed: 2018-06-10.

[14] *Flutter - Beautiful native apps in record time*. https://flutter.io/. Accessed: 2018-06-06.

[15] Brian S Hall. *Net Nanny for Android Review*. https://www.tomsguide.com/us/net-nanny-for-android,review-2754.html. Accessed: 2018-03-02.

[16] *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator*. https://w1.fi/hostapd/. Accessed: 2018-06-06.

[17] *HowTo dnsmasq*. https://wiki.debian.org/HowTo/dnsmasq. Accessed: 2018-06-09.

[18] *Inappropriate Content*. https://www.internetmatters.org/issues/inappropriate-content/. Accessed: 2018-05-30.

[19] Amanda Lenhart et al. *Teens, social media & technology overview 2015*. Pew Research Center [Internet & American Life Project], 2015.

[20] Andrew Lockhart. *Network Security Hacks*. " O'Reilly Media, Inc.", 2004.

[21] Ben Moore and Neil J. Rubenking. *Net Nanny*. https://www.pcmag.com/article2/0,2817,2467479,00.asp. Accessed: 2018-02-28.

[22] Ben Moore and Neil J. Rubenking. *Qustodio*. https://www.pcmag.com/article2/0,2817,2473800,00.asp. Accessed: 2018-02-28.

[23] *mux - Gorilla, the golang web toolkit*. http://www.gorillatoolkit.org/pkg/mux. Accessed: 2018-06-06.

[24] *Net Nanny Features*. https://www.netnanny.com/features/. Accessed: 2018-05-31.

[25] *netfilter/iptables project homepage - The netfilter.org project*. https://www.netfilter.org/. Accessed: 2018-06-06.

[26] UK OfCom. *Children and Parents: Media Use and Attitudes Report*. 2017.

[27] *Parental Controls and Internet Filtering - Circle with Disney*. https://meetcircle.com/. Accessed: 2018-04-28.

[28] *Parental controls and Privacy settings guides*. https://www.internetmatters.org/parental-controls/. Accessed: 2018-05-30.

[29] David Plummer et al. "An Ethernet address resolution protocol (RFC 826)". In: *Network Working Group* (1982).

[30] Justin Pot. *Adblock Everywhere: The Raspberry Pi-Hole Way*. https://www.makeuseof.com/tag/adblock-everywhere-raspberry-pi-hole-way/. Accessed: 2018-06-09.

[31] Vivek Ramachandran and Sukumar Nandi. "Detecting ARP spoofing: An active technique". In: *International Conference on Information Systems Security*. Springer. 2005, pp. 239–250.

[32] *Raspberry Pi - Teach, Learn, and Make with Raspberry Pi*. https://www.raspberrypi.org/. Accessed: 2018-03-01.

[33] *RFC 7519 - JSON Web Token (JWT)*. https://tools.ietf.org/html/rfc7519. Accessed: 2018-06-06.

[34] Neil J. Rubenking. *Circle With Disney*. https://www.pcmag.com/review/344240/circle-with-disney. Accessed: 2018-03-01.

[35] Mehran Sahami et al. "A Bayesian approach to filtering junk e-mail". In: *Learning for Text Categorization: Papers from the 1998 workshop*. Vol. 62. 1998, pp. 98–105.

[36] Jacob Salmela. *Block Millions Of Ads Network-wide With A Raspberry Pi-hole 2.0*. https://jacobsalmela.com/2015/06/16/block-millions-ads-network-wide-with-a-raspberry-pi-hole-2-0/. Accessed: 2018-03-01.

[37] Jacob Salmela. *FTLDNS™ and Unbound Combined For Your Own All-Around DNS Solution*. https://pi-hole.net/2018/06/09/ftldns-and-unbound-combined-for-your-own-all-around-dns-solution/. Accessed: 2018-06-09.

[38] Jacob Salmela. *Seven Things You May Not Know About Pi-hole*. https://pi-hole.net/2017/05/12/seven-things-you-may-not-know-about-pi-hole/. Accessed: 2018-06-09.

[39] Vijay Sarvepalli. *DNS Blocking: A Viable Strategy in Malware Defense*. https://insights.sei.cmu.edu/sei_blog/2017/06/dns-blocking-a-viable-strategy-in-malware-defense.html. Accessed: 2018-03-02.

[40] *Setting up a Raspberry Pi as an access point in a standalone network (NAT)*. https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md. Accessed: 2018-06-06.

[41] Wonsun Shin and Nurzali Ismail. "Exploring the role of parents and peers in young adolescents' risk taking on social networking sites". In: *Cyberpsychology, Behavior, and Social Networking* 17.9 (2014), pp. 578–583.

[42] *SQLite Home Page*. https://sqlite.org/index.html. Accessed: 2018-06-06.

[43] Alicia Marie Tan. *Circle with Disney gives parents full reign over their kids' devices*. https://mashable.com/2015/11/06/circle-with-disney. Accessed: 2018-03-01.

[44] *The Go Programming language*. https://golang.org/. Accessed: 2018-06-06.

[45] Paul Vixie and Vernon Schryver. "DNS response policy zones (DNS RPZ)". In: *ISC Technical Note Series* (2010).

[46] Pamela Wisniewski et al. "Parental Control vs. Teen Self-Regulation: Is there a middle ground for mobile online safety?" In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM. 2017, pp. 51–69.