A Simple Approach to Develop a Parental Control via a Monitoring Home Network

Network Design and Computer Management, 120/Computer Science and Engineering

2019-02-21
Adekunle Matthew Adekolu, Lei Chen
Prefaces

This project has been another practical way to develop our skill and put what we have learned at the university to practice; it definitely improved our Perl skills. We would like to thank Nicolina Manson for the project idea and give big thanks to our supervisor Mahboobeh Parsapoor who always finds time to answer our questions. We would like to thank our friends for their references and supports regarding what material to study, especially Julien Pinget and Josef.
ABSTRACT

This thesis explains how a simple parental control can be developed. The system aims to limit kids’ activities rather than entirely restrict those activities. However, current solutions are applying some filtering methods that prevent kids from playing games. In order to overcome this issue, this thesis presents a simple approach to implement a cost-effective parental control using low-cost hardware devices and open source applications.

The main advantage of the suggested method is that it can be used without installing any additional software and gives parents general information about their kids' activities on the internet without preventing kids from playing online games.
# TABLE OF CONTENTS

1. INTRODUCTION ......................................................................................................................... 1
   1.1 Motivation .......................................................................................................................... 1
   1.2 Problem Definition ............................................................................................................. 2
   1.4 Outline ............................................................................................................................... 2

2. BACKGROUND ............................................................................................................................. 3
   2.1 Network Monitoring for Company ..................................................................................... 3
      2.1.1 Network Monitoring Tools for Companies ................................................................. 3
         2.1.1.1 NetFlow ............................................................................................................... 3
         2.1.1.2 Tcpdump .......................................................................................................... 4
         2.1.1.3 Wireshark ...................................................................................................... 5
      2.1.2 Network Monitoring for Company Security ................................................................. 6
   2.2 Network Monitoring for Home Security ............................................................................ 7
      2.2.1 Parental Control Software ......................................................................................... 7

3. METHODOLOGY OF THE THESIS ............................................................................................. 9
   3.3 A Simple Implementation ................................................................................................. 9
      3.3.2 Raspberry Pi ............................................................................................................. 13
      3.3.3 Script for Analyzing ................................................................................................. 15
      3.3.4 Send Report to User ............................................................................................... 17

4. IMPLEMENTATION .................................................................................................................... 18
   4.1 Scenario A ......................................................................................................................... 18
   4.2 Analysis A ......................................................................................................................... 18
4.3 Result A .........................................................................................................................20
4.4 Scenario B .....................................................................................................................23
4.5 Analysis B .....................................................................................................................23
4.6 Result B .........................................................................................................................24

5. Conclusions ..................................................................................................................26
  5.1 Discussion ..................................................................................................................26
  5.2 Future Works ..............................................................................................................27

References ...........................................................................................................................28

Appendix ................................................................................................................................30
LIST OF FIGURES

Figure 1: Netflow consists of four parts: Exporter, Collector, Storage and Analysis [5] ........... 4

Figure 2: Tcpdump command .................................................................................................. 5

Figure 3: Wireshark deals with the data .................................................................................. 6

Figure 4: Shows the simple implementation in a flowchart ..................................................... 9

Figure 5: Designed topology .................................................................................................. 10

Figure 6: Verifying tcpdump version ....................................................................................... 12

Figure 7: Assign a static IP address to Raspberry Pi ................................................................. 14

Figure 8: Connect to Raspberry Pi ........................................................................................... 15

Figure 9: Verified Perl ............................................................................................................. 15

Figure 10: Getting started with CPAN ..................................................................................... 16

Figure 11: Capturing all HTTP traffic ....................................................................................... 19

Figure 12: Passing Data to Script ............................................................................................. 20

Figure 13: Graphic Results ....................................................................................................... 21

Figure 14: Results the user received via their email ................................................................. 22

Figure 15: Result with only one IP address ............................................................................. 23

Figure 16: Results in pie chart form ......................................................................................... 24

Figure 17: Specific Device Result ........................................................................................... 24
ABBREVIATIONS

AIX: Advanced Interactive eXecutive
BSD: Berkeley Software Distribution
CPAN: Comprehensive Perl Archive Network
DHCP: Dynamic Host Configuration Protocol
DNS: Domain Name System
HP-UX: Hewlett-Packard Unix
HTTP: Hypertext Transfer Protocol
SSH: Secure Shell
SOHO: Small Office/Home Office
SSID: Service set identifier. is the name that is used to identify the network
TCP/IP: Transmission Control Protocol (TCP) and the Internet Protocol (IP)
WPA2: Wi-Fi Protected Access II
1. INTRODUCTION

As technology is evolving, different game consoles and devices are connected to the home network, providing fun activities and interesting games which may be educative and creative but sometimes destructive. The imperfect human mind cannot keep up and adapt as fast as the development of technology, hence, there is a big need for parental control. In this thesis, we compare different methods for parental control and suggest a simple resolution to monitor a home network based on hardware. Our solution is more effective than the conventional ones as it considers both children’s and parents’ purposes of using the internet.

1.1 Motivation

Research from Brainturk [1] shows that brain functions can be improved by playing games because games boost memory but also enhance problem-solving skills. There are so many online games aiming at increasing the strategic ability of kids and motivation by playing strategy, board and trivia games. However, current parental control applications employ filtering content to restrict children from playing games. The parents’ purpose when installing a parental control system is not to totally forbid their children to play games. Is there a better way instead of filtering content so that children can play games under parental control?

This motivates us to implement a monitoring home network as a parental control. This solution is not only low cost, user-friendly, and hardware-based but also more efficient than the conventional solutions which focus on content filtering. Most of those solutions are based on software which must be installed on a computer or smartphone device and run in the background. The contents of a website do not mean that it is either malicious or abusive. Nevertheless, the databases of current solutions maintain bunches of segregated words which relate to the content of the website. This means that harmless websites containing keywords in the filtering database will be blocked.
1.2 Problem Definition

In order to provide an effective parental control system based on hardware, the solution must fulfill the following questions:

1) How could a cost-effective and independent simple parental control be implemented?
2) How can the proposed method prevent the kids using any external proxy to bypass the parental control?
3) How can a daily report be mailed to the parents without navigating to some external sites?
4) How can parents monitor the browsing history of the kids on a computer or smartphones without using an external server knowing about their internet usage?

The answers to these questions combine cheap hardware and open source software to implement a simple parental control. The detail will be shown in section 3 and 4.

1.4 Outline

This thesis will represent a suggestion for implementing a simple parental control by monitoring home networks. After this introduction, in section 2, we discuss the background of the project. Section 3 suggests a method for implementing a simple parental control by monitoring the home network. Section 4 discusses the results and analyzes the method as suggested in section 3. Finally, section 5 concludes the thesis and discusses contributions and possible future works.
2. BACKGROUND

2.1 Network Monitoring for Company

This chapter considers monitoring an organization network with the existing monitoring tools. It gives brief descriptions of how the tool works and why it is essential for every organization to keep track of their network usage.

Network monitoring has been an essential part of modern company networks. Network monitoring is also about collecting data which provide the necessary information for the network management. The purpose of network monitoring is the aggregation of useful information from various parts of the network so that the network can be managed and controlled using the collected information [2]. The network has already become an essential part of the computer. It is becoming increasingly important to make the network more efficient and locate network problems quickly.

2.1.1 Network Monitoring Tools for Companies

To deal with network monitoring and analysis, there are a variety of tools such as the Simple Network Management Protocol (SNMP), Tcpdump, Wireshark, Sniffing, and Network flow monitoring and analysis. Network behavior, such as application and network usage, utilization of network resources, and network anomalies and security vulnerabilities, can be analyzed based on the data packet and network traffic flow information [3].

2.1.1.1 NetFlow

Cisco Systems provides an open but proprietary network protocol running on Cisco IO (Internetworking Operating System), "Cisco NetFlow", in order to capture network traffic flow information and then send it back to the monitoring hosts [4]. NetFlow is used to analyze network traffic which can determine where the source is, where the destination is and how much traffic is being generated. NetFlow has three components: Exporter, Collector, Storage, and Analysis. Figure 1 below shows the operation of Netflow using the Cisco device. The Netflow
collector is connected to the network device and the packet is exported to the collector. Admin is connected to the collector via a console application which allows analyzing the collected packets.

![Netflow Diagram](image)

**Figure 1:** Netflow consists of four parts: Exporter, Collector, Storage and Analysis [5].

### 2.1.1.2 Tcpdump

Tcpdump is a powerful command-line packet analyzer that works on Unix-like operating systems and provides many features and options. A description of the contents of packets is printed out on a network interface that matches the Boolean expression. This description is preceded by a time stamp that, by default, is represented as hours, minutes, seconds, and fractions of a second since midnight [6].

For the better performance of a network, Tcpdump just captures the packets and saves them in a raw file. Although there are not so many analysis functions, many analysis tools are built for Tcpdump due to its feature. **Figure 2** below shows the basic tcpdump utility. For instance, a Perl script called "tcpdump2ascii" is used to convert the output from "tcpdump" raw file to ASCII format; "Tcpshow" is another utility that prints the raw "tcpdump" output file in human-readable format; "Tcptrace" is a free powerful analysis tool for "tcpdump" [6]. Different
types of results can be produced, e.g. results can be bytes, segments, elapsed time and received retransmissions, round trip times, window advertisements, and throughput.

Figure 2: Tcpdump command

2.1.1.3 Wireshark

Wireshark is an open-source packet capturing and network analyzer, previously known as Ethereal. Wireshark allows the user to see the network traffic flowing into and out of his/her windows or UNIX machine. It is used for troubleshooting networking problems but it is also an excellent way to learn exactly how the network protocols work. It runs on most computing platforms including Windows, OS X, Linux, and UNIX. Moreover, network professionals, security experts, developers, and educators around the world use it regularly [7]. Wireshark and Tcpdump are very similar. Both can capture and analyze network traffic on a different platform. Tcpdump is quite convenient for capturing data, but Wireshark’s interface is more user-friendly. The best practice is to use tcpdump for data capturing and Wireshark for data analyzing. Figure 2 shows a result of using Wireshark to analyze packet captured data.
2.1.2 Network Monitoring for Company Security

Due to the complexity of modern networks, network monitoring tools are essential parts of most companies. When critical resources like servers and applications are down or underperforming, network monitoring tools can play a critical role for the companies. For the normal days, they are also very helpful to keep the company service running smoothly.

The companies can benefit directly from network monitoring tools in various ways. For instance, many events that these tools monitor are symptoms of security events, e.g. attacker activity or malicious website. Network monitoring tools not only can help add security value but they are cost-effective.

Network monitoring products can help in security since many of these come with features that assist in data collection, host discovery and inventory [8].
2.2 Network Monitoring for Home Security

Parental control is a feature included or added to monitor and control the usage of a home network by tailoring what is important to see and to know and to present this information in a human-readable format. Parental control can be used at different stages depending on why it is important to have a parental control solution. Parental controls can be roughly divided into four categories: content filters, usage controls, computer management tools and monitoring.

**Content filters:** this is the traditional method for parental control. This kind of control ranges from putting a restriction on electronic appliances to using some software to control, restrict, or filter the contents that can be accessed on a network;

**Usage control:** this constrains the using of a device per user or limits which website they are able to visit, or it can forbid some certain usage like access restriction by not being able to download a new application like a proxy; in this case, it will require administrative privileges to do so. This can also be bypassed using exploit techniques.

**Computer usage management:** it lets the parent regulate the time kids spend on learning and playing games. The kids will be rewarded with more time to play games based on the time they study. Thus, the more time they spend on studying, the more time they have to play games;

**Monitoring:** this approach is mostly found in enterprise environments where the usage of the network can be monitored, i.e. the administrator is able to find out who is doing what and when. It mostly uses hardware resources to monitor all the devices that are connected to the network. Every time a new device joins the network it logs the information of the device and the activities for immediate or future analysis.

2.2.1 Parental Control Software

There are so many applications, both commercial and license, which use the concept of content filtering, usage control and other parental control categories as shown in table 1 below. In this thesis, we will introduce a few of these:

**Net Nanny:** this is one of the popular content filtering solutions. It can be managed remotely and is a good reporting tool. However, it needs to be installed on one of the computers in the home network.
**K9**: this is another popular solution specifically for protecting children. It is also strictly based on content filtering. Moreover, it has an anti-malware solution. It needs to be installed on one of the computers. It has a user interface to be accessed.

**OpenDNS**: this uses Domain Name System (DNS) to filter content on home networks. It is free but limited to content filtering. OpenDNS provides some recursive name server addresses.

### Table 1: Parental Control Software

<table>
<thead>
<tr>
<th>Tools</th>
<th>Price</th>
<th>Platform</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Nanny</td>
<td>39.99</td>
<td>Computer</td>
<td>Content filtering</td>
</tr>
<tr>
<td>K9</td>
<td>Free</td>
<td>Computer</td>
<td>Content filtering</td>
</tr>
<tr>
<td>OpenDNS</td>
<td>Free</td>
<td>Router</td>
<td>DNS filtering</td>
</tr>
</tbody>
</table>
3. METHODOLOGY OF THE THESIS

Controlling internet activities is about monitoring networks rather than using some form of traditional methods, which use applications to filter content web contents. The traditional method sets some limitation for which websites children can visit on the home network. In addition, it sets limitations to parent relationships with the children because the parents will not know anything if their children bypass the filtering instruction with some free proxy server. To effectively monitor a home network for the parental control purpose, there is a need to intercept the traffic on the network by using some open source tools. The tools can be used to capture and analyze network traffic based on how important it is for the parent to receive information. To achieve this aim, this thesis uses a cheap single board computer and open source tools for a simple approach.

3.3 A Simple Implementation

![Flowchart](image)

**Figure 4**: Shows the simple implementation in a flowchart
This simple implementation is based on hardware and open source software. It assumes that the user is using:

a. TP-Link TL-WR841N 300Mbps Wireless-N Router, a router that supports firmware upgrades.
b. Raspberry Pi 3 - Model B
c. 4GB SD card for raspberian image installation
d. 5V 4A Power Supply Adapter for Raspberry Pi
e. Ethernet cable

The design topology for simple implementation is shown below in Figure 5. Raspberry Pi is connected to the home router to monitor and capture the network using tcpdump. A laptop, iPad, and iPhone are connected to the wireless router. A work station is remotely connected to the Raspberry Pi using SSH.

Figure 5: Designed topology
The firmware of the home router needs to be changed by using OpenWrt in order to meet the goal of this implementation.

OpenWrt is a flexible operating system which is open source and is based on a Linux kernel primarily built for routing network traffic on embedded devices. The motivation behind using OpenWrt as the new firmware in this thesis is that it is easy to install. It has great security features which can be mostly found in enterprise routers. It allows packet capturing and analyzing. In addition to the above characteristics, it has one important feature, namely dynamic packet filtering (it checks the outgoing IP packets from the computer and allows incoming packets to get through the packet filtering if the packets are from the same computer as outgoing packets [9]).

### 3.3.1.1 Problems Installing OpenWrt

Initially, this implementation was started with D-LINK DIR 615 version H2 router. This type of router was supported by OpenWrt. Some technical problems of the suggested approach are as follows: issue while trying to change the firmware, namely the inability to access Luci (the web interface for configuring OpenWrt). This problem arose because version H2 does not support Luci.

### 3.3.1.2 How to fix OpenWrt issues

There are different approaches to resolve this issue but the quickest solution during this implementation was to change the router to TP-LINK TL-WR841N. However, the OpenWrt website forum suggests connecting the first router (D-LINK DIR 615 version H2) to OpenWrt through Telnet. Installing a web interface (Luci) is one approach to address the problem that was encountered during this implementation.

Now the router is up and running with the new firmware. By default, OpenWrt wireless is disabled, so we enable it by navigating to the default IP address 192.168. x. x on the web.
browser. 192.168.1.0/24 is the default network address for OpenWrt and 192.168.1.1 is the default gateway. This can be changed to any private network address. Create a Service set identifier (SSID), is the name that is used to identify the network in this project is called “OpenWrtthesis”. Wi-Fi Protected Access II (WPA2) is used for authentication because it is more secure and is currently the most secure protocol to protect wireless internet access. To connect to the router using SSH from the local machine, the command line can be used or any SSH client like putty to verify that the packet analyzer is installed. Figure 6 below shows the default installation of tcpdump without any configuration.

Figure 6: Verifying tcpdump version

Tcpdump is a command line tool for monitoring and analyzing a network, which allows the user to intercept the TCP/IP and other packets transmitted over the network. It is open-sourced and distributed under Berkeley Software Distribution (BSD) license and it works on almost all UNIX operating systems. Tcpdump for windows is windump, which uses wincap, a windows port of libcap.
3.3.2 Raspberry Pi

A credit card sized motherboard computer can allow different things. Interesting things can be done with this small computer on a Raspberry Pi website. The basic idea behind it was to improve the teaching of basic computer science. In this implementation, Raspberry Pi is used for the purpose of analyzing packets that are intercepted from the network by tcpdump. Raspberry Pi best fits this implementation because it is cheap and easy to set up. Furthermore, its operating system is free and is based on Debian, a Linux distribution that optimizes the hardware.

Setting up the Raspberry Pi for sniffing and analyzing requires loading it with a new operating system or using the default one. A quick setup guide on how to change the operating system can be found on the Raspberry Pi website. Because this implementation uses the default operating system Raspbian, we connect the Raspberry Pi to power and connect it to the router using a crossover Ethernet cable. An IP address is leased from the Dynamic Host Configuration Protocol (DHCP) automatically. First, we checked the router to find which IP address is assigned to it. Then, we navigated to the default gateway of the router, clicked on the status overview, and scrolled down to DHCP Leases. On the list of IP address leased to devices in this implementation, the IP address of Raspberry Pi was 192.168.1.157. This IP address should be stable. If a DHCP server is not renewed before the lease expires, it is most likely to lease the IP address to another device which will interrupt the Raspberry Pi. We assigned a static IP address to Raspberry Pi and excluded the IP address from the DHCP pool because we want this IP address to be permanently assigned only for Raspberry Pi. Figure 7 verifies the default state of the Raspberry Pi setting, i.e. obtaining an IP address automatically from the DHCP server and how to assign a static IP address to it. We use “$ifconfig” to check the network information for Raspberry Pi.
Figure 7: Assign a static IP address to Raspberry Pi

We connect to Raspberry Pi from a local machine or any SSH client like putty to make sure that the configuration is working and confirm that the working environment is ready to go. Figure 8 shows the version of Linux distribution that the hardware is running, more information about it and where you can find documentation. It also shows which network you are connected to. In this implementation, it is connected from elemOS.LAN, which is a machine on a local area network called elemOS. In this implementation, Raspberry Pi is connected using SSH.
3.3.3 Script for Analyzing

In this implementation, Perl is the programming language used to analyze the captured data. Perl is a high level, dynamic programming language suitable for data analysis and it also does many useful things. With Perl, it is even possible to write a new packet sniffer from scratch rather than using tcpdump. In this thesis, it is only used to analyze the data captured by tcpdump. Perl is pre-installed on every Linux distribution by default; “$dpkg –s perl” can be used from the console to verify the presence of it. Figure 9 shows which version of Perl is used by the current researchers.

Figure 8: Connect to Raspberry Pi

Figure 9: Verified Perl
Perl has rich modules which can do a lot of things. It can be downloaded from the Comprehensive Perl archive network (CPAN). Type “$perl -MCORE -e shell”. Figure 10 demonstrates how to get started with CPAN by downloading and installing the module.

![Figure 10: Getting started with CPAN](image)

This script uses the modules in the table below to write a script that is used to analyze the data captured with tcpdump.

### Table 2: Perl modules and description

<table>
<thead>
<tr>
<th>Modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net::TcpDumpLog</td>
<td>Used to read tcpdump/Pcap files</td>
</tr>
<tr>
<td>NetPacket::Ethernet</td>
<td>Used to decode ethernet packets</td>
</tr>
<tr>
<td>NetPacket::IP</td>
<td>Used to decode internet protocol (IP) packets</td>
</tr>
<tr>
<td>NetPacket::TCP</td>
<td>Decoding transport control protocol (tcp) packets</td>
</tr>
<tr>
<td>Time::HiRes</td>
<td>Used to calculate the duration of time</td>
</tr>
<tr>
<td>Data::Dumper</td>
<td>Used for printing data in an identical copy</td>
</tr>
<tr>
<td>Net::Whois::IANA</td>
<td>Extract WHOIS data</td>
</tr>
</tbody>
</table>
The full Perl script for this thesis can be found in Appendix A with comments. It can be copied and pasted but beware of syntax error. The code can be extended to do more sophisticated things.

### 3.3.4 Send Report to User

The idea of this simple parental control by monitoring the home network is to be able to run it for twenty-four hours a day or during specific hours without any further configuration by the user. To accomplish this, the bash script in Appendix B is used. Put this script in contrab in Raspberry Pi, control in unix by putting files that contain instructions to be executed. More about contrab can be found on the main page. The script runs tcpdump to capture data every six hours. Then it uses a sleep function to put the tcpdump command on hold while it analyzes the captured data. Once the data is analyzed, it resumes capturing data by overwriting the previous data. Consequently, no raw data is saved, which is a good way to free space.
4. IMPLEMENTATION

This chapter presents the result of the data that was captured and analyzed with Perl script within the specific scenarios and assumptions below. This implementation examined two different scenarios as described below and compares which one is more optimal in the concluding section of the dissertation. Both scenarios do have their advantages and disadvantages, which will be discussed in the last section. Generally, we assume that the parents trust their kids to decide which website they are visiting. This project can be further expanded to address the issue of pop-up websites like advertising banners, which is very common when visiting or navigating through websites. This project can also be expanded to do more sophisticated network monitoring and analysis.

4.1 Scenario A

In this scenario, the home network consists of three devices: a smartphone running Android and two computers – one is running Linux and the other one is running window 8.1. We assume that the general assumption is implied; thus, we presume that the general premise is meant and the user wants to supervise the whole network. Understand which devices were connected, and what website were visited and for how long, also to find out if any device joins the network. The network is supervised for three hours and thirty minutes in this implementation but is not restricted to this fourth dimension.

4.2 Analysis A

Based on scenario A, data is captured using tcpdump and analyzed with the Perl script. The scenario above shows that the user desires to find out the web traffic on the network using options in tcpdump to limit what data to capture. This is because tcpdump has many options and this implementation will not spill the beans about all the options that tcpdump provides, but the options are on the tcpdump main page, which can be found in Appendix C. This implementation will explain which options are passed to tcpdump to capture the data. Create a working directory with the command “$mkdir thesis” from the command prompt and change to this directory using “$cd thesis”. Everything will be saved in this directory including the Perl script. Connect to the
router using SSH from the present working directory and run the `tcpdump` command on the router. The following options are passed to the `tcpdump`:

- **n**: tells `tcpdump` not to translate the host address to the hostname.
- **i**: specifies the interface that is to be monitored. OpenWrt uses bridged br0 or br-lan to make multiple physical network interfaces act as if they were just one interface. We need to monitor this interface to be able to capture all the traffic on the network. Another option to monitor all the traffic without using SSH is to use a feature in OpenWrt called `iptables` to mirror all the traffic to the Raspberry Pi interface. Please see Appendix D for this option which will send a copy of all the traffic to a specified address.
- **s**: specifies the length. By default, `tcpdump` captures the full length but for this implementation, only packets from the data layer are needed, so specifying 1556 will be enough.
- **port not 22**: this option tells `tcpdump` not to sniff anything on SSH because there is no need for it.
- **port 80**: this option tells `tcpdump` to capture only Hypertext Transfer Protocol (HTTP) traffic because that is the default port number for web traffic.
- **w**: this option tells `tcpdump` to save the result in a file called “http_traffic_all_devices” in this implementation.
- “>” redirects it to save on the present Raspberry Pi working directory. The memory on the router is not enough to save such huge data. Figure 11 shows how to connect to OpenWrt via SSH and capture data.

```bash
pi@hhthesis ~/thesis $ ssh root@192.168.1.1 tcpdump -ni br-lan -s 1556 -w - > http_traffic_all_devices
```

**Figure 11: Capturing all HTTP traffic**

Figure 12 below shows how data capture is being passed using the file name to script for execution.
The Perl script will read the tcpdump file and analyze the data by running "$perl version1.0.pl", assuming that they are in the same directory. In this implementation, it takes about six minutes to analyze this data. Raw data using the tcpdump read option can be found in Appendix E.

4.3 Result A

The user is more concerned about which devices visit which websites and for how long, so the time stamp, the source computer and the destination IP address in this scenario are the most important aspects. The image below depicts the virtual result in a bar chart. A simple python script is used to draw this chart. The script formats the data and then uses python “matplotlib.Pylot”, a library in python that is able to define and plot a graph. The python script can be found in Appendix F. Using python to draw the chart is not the goal of this thesis; it is just for visualization. Figure 13 shows the first ten lines of the data using bar charts.
In the above Figure 13, please note that the result is huge, so only the top ten lines are indicated by using “head -10”. While reading the result to show the first 10 lines, the Y-axis is the time in seconds and the X-axis is the website IP address. The color indicates which home device goes to which website and for how long.

Sending a report to the user to know what is happening on the network in a user-friendly format is the goal of this implementation. The best way to do this is to use the mail utility in Linux to send electronic email to the user. This is best because most people know how to access their email and most people do so at least once a day. We then configure the mail utility in Raspberry Pi to use the GMail SMTP.

In this implementation, we use the command $perl version1. 0. pl | mail –s” Alert! Daily Network Activities’ Report” useremail@domain.com to send the result. This command is included in the sendmail script because needs to be run every specific hour. What happens is that instead of printing the result to the screen, it uses the pipe option and mail utility configured in Raspberry Pi to send a notification to the user’s email address. Figure 14 below shows the information a user received via their email address.
As shown in figure 14, there are three columns. The first column shows how long a computer has been on a website, the second column shows which computer, and the third column shows which website. Note that this is only a small excerpt from the result that the user received. Also note that the Destination Description should be a host address, for example, like www.google.com. In this implementation, it was hard to stripe out the host name or to convert the IP addresses to the host address. One IP address sometimes map to multiple host, use the WHOIS module in Perl to search for the description of the destination instead of presenting numbers to the user, as in Figure 15 below.
4.4 Scenario B

In this scenario, the network consists of the same devices and the user knows which website is to be monitored. Two websites are to be monitored and a report sent to the user via email. Assuming that the general assumption is meant, they are interested in usage per device that signifies how long each device spent on each site respectively. The data will be captured for three hours and thirty minutes in this implementation.

4.5 Analysis B

The procedure in this analysis is the same; only the tcpdump option needed to be redone to match the demand of the scenario. Reconstruct tcpdump syntax because tcpdump is flexible with many options. The next step is to specify the destination host by typing the following command on Raspberry Pi: “ssh root@192.168.1.1 tcpdump -ni br-lan -s 1556 'host www.youtube.com or host www.miniclip.com' and 'tcp port 80' and 'port not 22' -w - > youtube_miniclip”. In this command, there is a new option – “host” and new filename, which tells tcpdump to capture only traffic arriving or departing from the specified hostname and save it with the filename “youtube_miniclip”. Any other traffic will be ignored. To capture the data based on scenario B, we use the Perl script to analyze it the same way as in analysis A. However, the name needs to be changed to this new filename otherwise the script will go ahead and analyze previous data.
4.6 Result B

Based on the analysis in B, we derived the result in pie charts for virtualization using the python script. Figure 16 below shows the result in percentage.

Figure 16: Results in pie chart form

From the pie chart, the left side shows the percentage of users visiting one website. The right side of the chart shows which user visited which websites and their percentage. The same method in Result A is used to send report information to the user, namely the sendmail script. The date that each user will receive this information varies, as shown in Figure 17 below.

Figure 17: Specific Device Result
Figure depicts the total resolutions for the whole data captured in scenario B. The script is more optimal when the user knows what they want to see and from which device, as it takes less time to analyze and they also have reduced data in their electronic mail.
5. CONCLUSIONS

In this thesis, a simple parental control via monitoring a home network solution which is not based on a specific operating system was introduced. Moreover, there was no need to have additional software running on any personal device, but instead, we connected the Raspberry Pi to the router and monitored the activities. The monitoring information is periodically reported to the parents via email so that they can keep track of their kids’ activities even when they are not home. It provides a new direction to reach the children safely and enhance their learning and thinking ability. Different scenarios were examined and data was captured based on the scenarios. After analyzing both scenarios, it has been shown that scenario B is more efficient and less data is sent to the user.

Current parental control solutions just focus on restriction, e.g. using usage control software to block some websites. However, these approaches are not efficient and do not protect kids from some malicious websites because there are many ways that the children can bypass the current solutions without their parents’ awareness. We argue that providing more information about kids to their parents would be better than limiting the children’s accessibility to the network. Our solution does not block any websites; instead, it collects information for the parents. If the parents have more information about their kids, they will find a good way to address their children’s issues, not just stop them. Nevertheless, this implementation does not take off any pop-up link that may arise during children’s visit to a specific website. This can be a problem because a website with unwanted or inappropriate content may pop-up, but we assume that the children know which website they want to visit further.

5.1 Discussion

This thesis addresses the problem of installing client/server applications on devices to filter content. When the users do not filter content, there is no redirection of a webpage so no other external server is aware of their internet usage. With this presented solution, captured and analyzed data do not upload on any server, and everything happens within the network. It also provides features that are mostly found on enterprise level devices, which can be very expensive for home networks. This makes it possible to turn a home network into a small office network.
without buying an additional expensive router or server. However, it requires additional configuration, which is not implemented yet.

This suggested solution does not address the issue of unexpected websites that might pop up (issue of user click and pop-up) which can be superficial. Tcpdump captures mostly all the packets even though the size is specified. We have found issues with the memory; when the data is huge, it takes more memory and causes the router to be slow. Analyzing such huge data with the Perl script caused it to become unresponsive so we could not run the tcpdump for 24 hours as expected. Another issue was that we could not make the script automatically translate the device IP address to the hostname.

5.2 Future Works

This solution is a simple implementation for controlling online games in home networks. If, and only if, the assumption is meant, this implementation manually configures names for devices in the network. In case a new device or any device joins the network and the name is not manually configured in the script, it will mark as unknown and display its Mac address.

The script can be restructured to print the source IP address and translate the IP address to the hostname using “gethostbyaddress”, which is a function in socket programming. At the moment, the user’s email is manually configured so if the email is hacked, the user will not be able to have access to reports, which can cause a security threat.

This work can also be improved in the following direction:

- Provide a web interface which will allow the user to update their email address by one click and host this web on the suggested solution.
- Expand the functionality of OpenWrt to address the subject of any unexpected or unwanted pop-up website and also implement the firewall technology and security features like an intrusion prevention system, which provides some protection against botnet [10].
- Expand the functionality of Raspberry Pi to utilize it for a smart, intelligent device for the home by implementing a security alarm and a server for home networking.
REFERENCES


APPENDIX

APPENDIX A Perl script to analyze data

Appendix A is a script used to analyze the home network. The scripts use various Perl modules which can be downloaded from the comprehensive Perl archive network (CPAN). The commented area is donated with a comment sign which explains each module. To access any module in Perl, use “use” followed by the name of the module. Basically, what this code does is read data captured by tcpdump. It dissembles the IP/TCP then sorts out the important information like the site visited, how long and the mac address of the device.

# Copyright HH Thesis, 2014
#This script is used for analyzing Home network monitoring for parental control and can be extend to do more stuff

#Matthew
use strict;
use warnings;
use Net::TcpDumpLog;
#this module is used to read data and heads tcpdumps. need to install
use NetPacket::Ethernet;
# used to assembling or disassembling packets using their Ethernet protocol, which can be IP, TCP etc.
use NetPacket::IP;
# used in assembling and disassembling of internet protocol (IP) packets.
use NetPacket::TCP;

# used in assembling and disassembling of transmission control protocol (TCP) packets.
use Time::HiRes qw(gettimeofday);
# used to get the current time of the day
use Data::Dumper;
use Socket; # For Binary-> IP dot quad etc...
use Net::Whois::IANA; # need to install from cpan For IP -> Name
no warnings "uninitialized"; # neglect unwanted warning
use IO::Socket::INET;
use Text::Truncate; # need for cutting off the description text cos is too long and needed to be install from CPAN

my $TCPLOG = 'youtube_miniclIP';  # TCP Dump Log
my $log = Net::TcpDumpLog->new();
$log->read ($TCPLOG);  # Packet that I need to analyze

my %srcdest;

# Decoding packet and analysis starts here.

foreach my $index ($log->indexes) {
    my ($length_orig, $length_incl, $drops, $secs, $msecs) = $log->header($index);
    my $data = $log->data($index)
    my $eth_obj = NetPacket::Ethernet->decode($data); # Mac address
    next unless $eth_obj->{type} == NetPacket::Ethernet::ETH_TYPE_IP;
    my $mac_src = $eth_obj->{src_mac};  # Source
    my $mac_dest = $eth_obj->{dest_mac};  # Destination
my $IP_obj = NetPacket::IP->decode($eth_obj->{data});  # Analyze IP
next unless $IP_obj->{proto} == NetPacket::IP::IP_PROTO_TCP;
my $IP_src = $IP_obj->{src_IP};
my $tcp_obj = NetPacket::TCP->decode($IP_obj->{data});
my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime($secs +
$msecs/1000);
my $time = sprintf("%02d/%02d/%02d %02d:%02d:%02d.%02d", $mon, $mday,$year,
$hour, $min, $sec, $msecs);
my $port_src = $tcp_obj->{src_port};
my $IP_dest = $IP_obj->{dest_IP};
my $port_dest = $tcp_obj->{dest_port};
# Gather $mac_src to $IP_dest times
next unless defined $mac_src;
next unless defined $IP_dest;
my $sdkey = "$mac_src|$IP_dest";  # Key has $mac_src and $IP_dest
if (not defined($srcdest{$sdkey})) {  # New connection
  my @times = ( );  # List of times
  $srcdest{$sdkey} = \@times;
}
my $times = $srcdest{$sdkey};    # Append event time
push @$times, $secs + $msecs/1000;
}
# Sort by time (in secs)
sub byTime {
    return ($a <=> $b);
}

# Convert time difference to hr:min:sec
sub DiffConvert($) {
    my ($diffsec) = @_; 
    my $roundsecs = sprintf "%.0f", $diffsec; # Round to secs
    my $hr = $roundsecs / (60*60); # Hours
    my $remmin = $roundsecs % (60*60); 
    my $min = $remmin / 60; # Minutes
    my $sec = $remmin % 60; # Seconds
    return($hr, $min, $sec); # ($hr, $min, $sec)
}

print "===========================================================================

<table>
<thead>
<tr>
<th>Duration</th>
<th>Source Computer</th>
<th>DESTINATION DESCRIPTION</th>
</tr>
</thead>
</table>

while (my ($sdkey, $times,$hostname, $sock) = each %srcdest) {
    next unless scalar(@$times) > 1; # Insufficient dat
    my ($mac_src, $IP_dest) = split /\|/ , $sdkey; # Keys
    my @events = sort byTime @$times; # Order events
    my $late = pop @events; # Last event
my @difftime = DiffConvert ($late - $events[0]);

# look for IP address through iana database, and if the description is too long, cut it to 50 characters.
my $iana = new Net::Whois::IANA;
$iana->whois_query(-IP=>$IP_dest);
my $name = $iana->descr();
my $final_name = truncstr($name, 50);

# hard coded local device's mac address and we have just mask it out
my $lei2 = "88cb87cXXXXX";
my $matt = "bc77373XXXXX";
my $IPho = "7cc537XXXXX";
my $hhthe = "b827eXXXXXXX";
my $pc = "90fXXXXXXXXXX";
my $andri1 = "847XXXXXXXXX";
my $joakim = "90XXXXXXX";
my $lei = "c04aXXXXXXXXXXX";

# check which device has which mac address
if($lei eq $mac_src){
    $mac_src = "Lei";
}elsif($lei2 eq $mac_src){
    $mac_src = "Lei's IPad";
}elsif($matt eq $mac_src){
    $mac_src = "Matthew's PC";
elsif($IPho eq $mac_src){
    $mac_src = "Matthew's IP";
}elsif($hhthe eq $mac_src){
    $mac_src = "Monitor Device";
}elsif($joakim eq $mac_src){
    $mac_src = "Joakim Comp";
}elsif($andri1 eq $mac_src){
    $mac_src = "Mathew's Phone";
} else {
    $mac_src = "Unknown: $mac_src";
}

printf "| %d:%02d:%02d | %s-40s-%10s %s-53s%s |\n", $diff[0], $diff[1], $diff[2], $mac, $final_name;

print "="
print "="
print "="

APPENDIX B Bash script placed in contrab

Appendix A is used to capture data only when it runs. However, data needs to be captured based on a scheduled time using a bash script. The script below uses the
sleep function in unix to specify how long it will take to run. Every 900 secs, the script will execute the command to capture the data.

#Edit contrab and put this script

#by Lei

# this script is to schedule tcpdump to run every 6 hours

#!/bin/sh

sleep 900

    ssh root@192.168.1.1 tcpdump -ni br-lan -s 1556 -w -> http_traffic_all_devices

crontab command: * /6 * * * /path/ to /command

# update the file after every 6 hours

#!/usr/bin/perl

use strict;

use warnings;

my $file=" http_traffic_all_devices";

if(-e $file)

    {

        unlink(" http_traffic_all_devices.pcap");

    }

unless (open FILE,'>'.$file)

    {

        die"nUnable to create $file";

    }

close FILE;
#analyse the data with perl script every 6 hours while it sleeping for 15 mins and send result to user.

*/6*** /perl version1.0.pl | mail –s "Alert! Daily Network Activities’ Report"

useremail@domain.com

APPENDIX C Tcpdump options

This appendix contains a table of tcpdump options with explanations.

http://www.tcpdump.org/manpages/tcpdump.1.html

APPENDIX D Packets mirroring using IPtables

Route the gateway traffic through Raspberry pi.

IPtables -A PREROUTING -t mangle -j ROUTE --gw raspberry pi add --tee
IPtables -A POSTROUTING -t mangle -j ROUTE --gw raspberry pi add --tee

APPENDIX E Raw data read with tcpdump read option

This is raw data was captured using tcpdump while streaming YouTube on the network.

$ tcpdump -r youtube_miniclip

reading from file youtube_miniclip, link-type EN10MB (Ethernet)

14:56:28.209814 IP 192.168.1.168.38735 > fa-in-f93.1e100.net.http: Flags [S], seq 3826273272, win 14600, options [mss 1460,sackOK,TS val 18426486 ecr 0,nop,wscale 4], length 0
APPENDIX F Python script for virtualization

This script is used to virtualize the analyzed data.

#!/usr/bin/env python

#matthew
import re
import sys
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
def humanize(total_seconds):
    #print('debug: %s' % total_seconds)
    hours = total_seconds / 3600
    total_seconds -= hours * 3600
    minutes = total_seconds / 60
    total_seconds -= minutes * 60
    seconds = total_seconds
    return '%.02d:%.02d:%.02d' % (hours, minutes, seconds)
d_devices = {}
d_services = {}
d_combi = {}
for line in sys.stdin:
    res = re.search(r"(\d+):(\d+):(\d+)s+|s+(\[w :']+)s+|s+(\[w.]+)", line)
    hours, minutes, seconds, device_name, service_name = res.groups()
    total_secs = int(hours) * 3600 + int(minutes) * 60 + int(seconds)
    d_devices[device_name] = d_devices.get(device_name, 0) + total_secs
    d_services[service_name] = d_services.get(service_name, 0) + total_secs
    d_combi[(device_name, service_name)] = d_services.get((device_name, service_name), 0) + total_secs
#pie chart plotting

plt.suptitle("Usage per device")

keys = d_devices.keys()

plt.pie([d_devices[key] for key in keys], labels=keys, shadow=True, autopct='%.1f%%', explode=[0.05 for i in keys])

plt.show()

plt.suptitle("Usage per service")

keys = d_services.keys()

plt.pie([d_services[key] for key in keys], labels=keys, shadow=True, autopct='%.1f%%', explode=[0.05 for i in keys])

plt.show()

plt.suptitle("Usage per pair of device and service")

keys = d_combi.keys()

plt.pie([d_combi[key] for key in keys], labels=['%s > %s' % (i[0], i[1]) for i in keys], shadow=True, autopct='%.1f%%', explode=[0.05 for i in keys])

plt.show()

# Bar chart plotting

# Bar chart plotting

c_services = d_services.keys()

c_devices = d_devices.keys()

#c_pairs = d_combi.keys()
matrix[device][service] = time

matrix = {}

for pair, time in d_combi.items():
    device, service = pair
    matrix.setdefault(device, {})[service] = time

width = 0.9

from random import random

colorscheme = [(random(), random(), random(), 1) for i in c_devices]
bars = []

for i_device, device in enumerate(c_devices):
    bars.append(plt.bar(
        range(len(c_services)),
        [matrix[device].get(service, 0) for service in c_services],
        width,
        color=colorscheme[i_device]
    ))

plt.xticks([i+width/2. for i in range(len(c_services))], c_services)

plt.legend([p[0] for p in bars], c_devices)

plt.show()
My name is Lei Chen from China. I am pursuing a Bachelor of Computer Science and Engineering from Halmstad University. Now I am working on a start-up company from silicon valley called Zippia. My major responsibility is to collect, clean, and analyze data.

Adekunle is a young enthusiastic professional and a dynamic and hard-working Network engineer who is passionate about Network Communication and Security. He studied Network Design and Computer Management from Halmstad University, Sweden.