EagleEye
Wi-Fi Monitoring for Non-Technical Users

Pedro M. Sosa, Michele Haque, Oeyvind Reinsve

May 23, 2017

Abstract

The Internet's rapid expansion and the drop in networking equipment prices, has lead to the increase of personal wireless networks in households and small businesses around the world. However, many of these networks are being administrated by users non-technical users who lack strong networking knowledge. Because of this, many users are getting subpar network performance, or are suffering from easily detectable problems. In hopes of bridging this knowledge gap, we set out to develop EagleEye, a wireless monitoring tool specifically designed to arm non-technical people with the proper information to help them troubleshoot their own personal networks. To do this, we surveyed 813 users to understand their current networking knowledge, troubleshooting mechanisms, and network monitoring needs. Afterwards, we developed and released a modular and easily expansible, open source tool to empower non-technical network administrators.

1 Motivation

The Boom of Personal Networks Nowadays, the Internet has become a critical necessity. More and more people are requiring access to it, and networking equipment prices has become quite inexpensive. This has led to an increase in personal networks at homes or small businesses, the vast majority of which are administrated by non-technical users. In fact, by September 2016, it was estimated that in the U.S. alone, about 75% households had Wi-Fi networks [1].

While most routers and network devices intended for personal use attempt to be simple to set up, there is still a gap in knowledge that prevents most users from troubleshooting and fine-tunning their own networks. Because of this, many users are getting subpar network performance, or are suffering from easily detectable problems. We believe that to fill this gap, users need an easy to use network monitoring tool that could present important information in an accessible manner. It is with this idea in mind that we set to develop EagleEye, a wireless network monitoring tool for everyone.

Survey To better understand the needs of the users, we surveyed 813 (58.6% Male; 41.4% Female) users about their personal networks.

Our first questions were aimed at accessing the understanding of Wi-Fi setup and features. We found that 79.4% of the users interviewed had at some point set up a personal Wi-Fi network (either at their home, office, or somewhere else). Of those 30.4% had not renamed the access points or changed their default passwords, and 33% where unaware that they could change their router's broadcasting channel.

We then asked users about their troubleshooting techniques. 69.5% stated that they're preferred technique was to "turn the router on and off again", while 17.5% of users attempted to troubleshoot using the internal router's website (or other networking tools).

Finally, we briefly described the graphs and information that would be displayed on our monitoring application, and 76.5% of all users stated that they would find such tool useful for troubleshooting their wireless networks.

Goals EagleEye seeks to enable users to troubleshoot a variety of problems in their networks. The most common problem it seeks to help is performance issues. Performance issues can be particularly important to solve in environments with low bandwidth or poor Internet connectivity. For example, network misconfiguration in a household at a developing country could compound with the already poor connectivity in the area, and render the network essentially unusable.

Another problem we seek to prevent is insecure Internet access or malicious activity. In the last couple of years, fueled in part by the IoT movement, we have seen how wireless devices connected to the Internet have been hijacked by Bot-nets to carry out DDoS attacks [2]. We have also witnessed virus, worms, and Trojans that infect and exfiltrate data from personal computers. EagleEye can help catch any malicious activity be monitoring the usages of each device, the amount of data that is being transfered to/from each device, the protocols that are being used, and any sensitive data that might be getting leaked.

2 Related Work

Personal Network Performance Some studies such as [3], [4], and [5] have focused on measuring the performance of home networks. These researchers found that a large portion of home networks are not actually performing to the fullest extent. However, the cause of this is not necessarily well defined. In general they found that "while wireless links inside homes tend to be stable over time there is significant performance variation across links, and many links are highly asymmetric" [4]. In some cases, it seemed that variability in link quality was due to wireless interference due to either poor router placement, interference from other electronic devices, or over-usage of a specific wireless channel. In other cases, it seemed that minor client misconfiguration, or specific misbehaving clients would hamper the network. However, most of these studies pointed that the reasons for this poor performance could be varied, and there was a need to further monitor these networks.

Existing Networking Tools Wireless monitoring tools are not a new concept, as there are many such tools in the market today. Some of these tools, such as Wireshark [6], are extremely comprehensive and allow users to analyses networks on a deep per packet basis. Others such as DataDog [7], PRTG [8], LogicMonitor [9], and Spiceworks [10] provide strong monitoring features aimed at big businesses with large networks. However these types of monitoring tools require extensive networking knowledge, and offer features that are either too complex or simply unnecessary for personal networks.

Some companies, such as Asus, Linksys, and Cisco, provide simple network monitoring software inside their routers. However, these tools have two main drawbacks. First, since these tools come pre-bundled into their devices, it requires the user to own that specific device. This is overlooking the millions of networks that use networking equipment from other companies or older devices. The second issue, is that some of these tools are too simple, and only give you a very general overview of network utilization. This means that the user has very limited information to do simple troubleshooting in the network.

Monitoring Techniques Lastly, it is worth mentioning that there has been a lot of research done in the area of wireless monitoring techniques. Uma [11] presents efficient techniques for network monitoring based on different techniques (data driven, active, passive, and self-configuring monitoring among others). This paper provides a wealth of ideas, that could be adapted for many different situations. On our particular case, we are focusing on passive scanning, since it provides all the information we need, without directly interacting or modifying with the home network. Another set of interesting studies has been done focusing on surveying already existing monitoring tools. Some of these studies [12], [13] compare the features and benefits of tool used to monitor large business networks, such as FlowScan, Autofocus, and Fluxoscope. Although these two papers focus on large-scale networks, they provide a good starting place for an effective wireless monitoring design.

3 Implementation

3.1 Architecture

EagleEye is a monitoring tool that is device agnostic and does not require users to do any modification to their networking equipment. It is a portable software that sits on a separate computer which sniffs network traffic. Figure 1 describes EagleEye's modular design. This design architecture allow for easy code modification or further extension. It has been separated into a back-end, which monitor and analyses the packets, and a front-end, which presents and organizes the pre-processed data.

The current iteration of EagleEye can run on most Debian Linux distributions, however the entire program is designed in a modular fashion to allow for future development or porting onto other devices. It does require a dedicated wireless interface that can be placed in Monitor mode.

Back-End Since most networks today are secure with WPA2, our back-end leverages *dot11decrypt* [14] and *libtins* [15] (a C++ networking library) to decrypt our network's packets. These packets are then filtered and analyses using Python's networking library Scapy [16]. Finally all of this information is saved in a

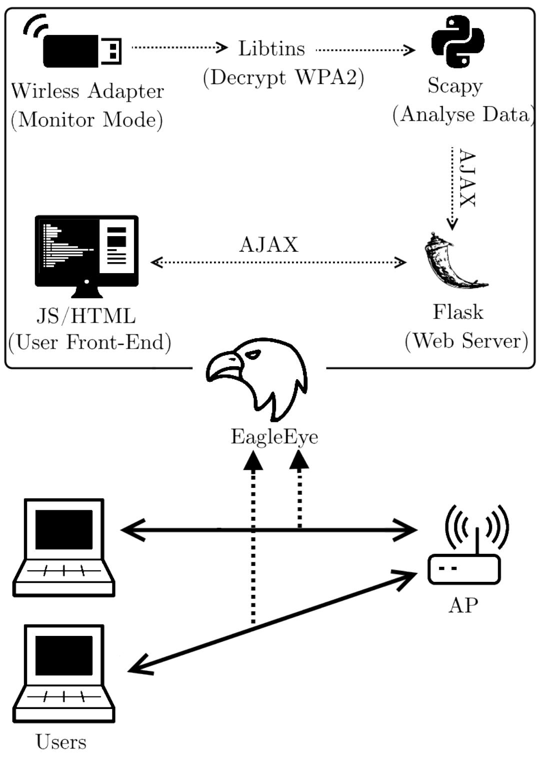


Figure 1: Implementation Structure and overview of technologies used.

Flask micro-server that will communicate with out front-end.

Front-End The graphical interface for EagleEye consists of a dynamically created website. This website is hosted within the same computer, and it can be accessed with any modern web browser. This site does frequent AJAX calls to the Flask server to retrieve any new information captured. The site is completely dynamic, meaning it can accommodate a fluctuating number of devices, and update quickly. We leverage Chart.JS to build visually appealing interactive maps to represent most information.

Packet Analysis An important architectural design was choosing the correct structure to save the monitoring information. As one can see in Figure 2, we decided to split the monitoring into time-slots (or windows) of 5 seconds. However, instead of choosing to keep a list of time-slots containing the network information of each client's activity, we choose a list of 'Client' structures. Each client structure contains the basic client information (IP Address, MAC address, OS, etc.) and also a 'report' structure. Inside the 'report' structure we keep the client's activity that happened inside a given 5 second time-slot. If a client is not active during a given 5 second time-slot, the information is not saved, thus saving us space.

This structure was specifically chosen because it decreases the amount of redundant or necessary information, so the overall structure is always quite small (e.i. we are only logging information whenever the client is active). However, the trade-off is that the front-end will need to spend more time arranging the data to properly build charts and calculate basic usage information. Nonetheless, the time needed for our front-end to run the necessary calculations is substantially smaller than the time needed for our front-end to retrieve a substantially bigger, yet organized, data-structure. Furthermore, we were interested in taking away as much complexity from the back-end as we did not want to risk loosing packets or introducing other irregularities on the packet capture.

3.2 Features

The EagleEye web interface (shown in Figure 3) is separated into two sections: The main aggregate section, and the clients section.

Aggregate Section The Aggregate section allows users to quickly view the overall usage of the network. It presents basic information such as the AP name, MAC address, broadcasting channel, and total amount of data uploaded/downloaded. It also presents three graphs:
- **Instantaneous Uplink & Downlink Throughput.** This graph is generated by calculating the aggregate amount of information that was transmitted to and from the router and dividing by the time it took to transmit that information.
- **Uplink and Downlink Usage.** This graph shows the aggregate total number of packets sent and received.
- **Dropped Packets**. This graph shows the aggregate number of duplicate (retransmitted) TCP packets.

Clients Section The Client section allows users to view the behaviors of a specific client in the network. It presents three graphs:
- **Usage** This graph shows the amount of packets sent and received by the client across time.
- **Packet Types** This graph shows the amount of TCP vs. UDP packets that a specific client has used across time.
- **Ports** This graph shows the top ports used by the client to communicate across time.

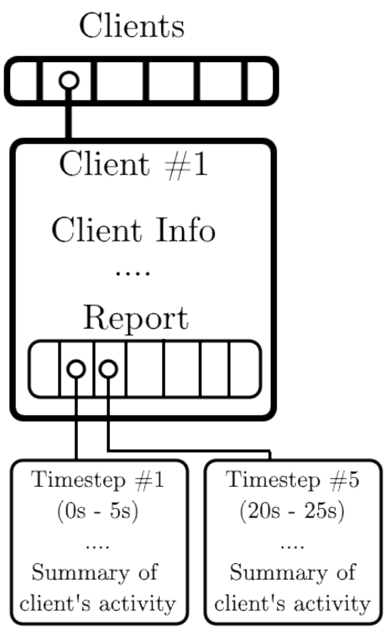


Figure 2: Example of the structure used to save the monitoring information. Notice how in this case, Client #1 has only been active during the 1st and 5th window (which correspond to time 0s to 5s, and time 20s to 25s)

3.3 Usage

One usage option, would be for users to set up EagleEye on a small device near the access point, and allow it to run as they themselves use the network normally. Then, upon finding an issue with the network, they could refer to the EagleEye's logs to view and better understand the issue that they are facing. Alternatively, our software could be used reactively, activating it only when issues are encountered.

4 Testing & Characterization

Since the goal of EagleEye was to create a tool that would empower non-technical users to troubleshoot their personal networks, we focused our testing on proving the actual accuracy and responsiveness of the tool. We also believe that it would be imperative to do further user testing, because although EagleEye was built using the initial feedback obtained by surveys and the previous research done on home networks, the success of this tool depends on continuous engagement with the users.

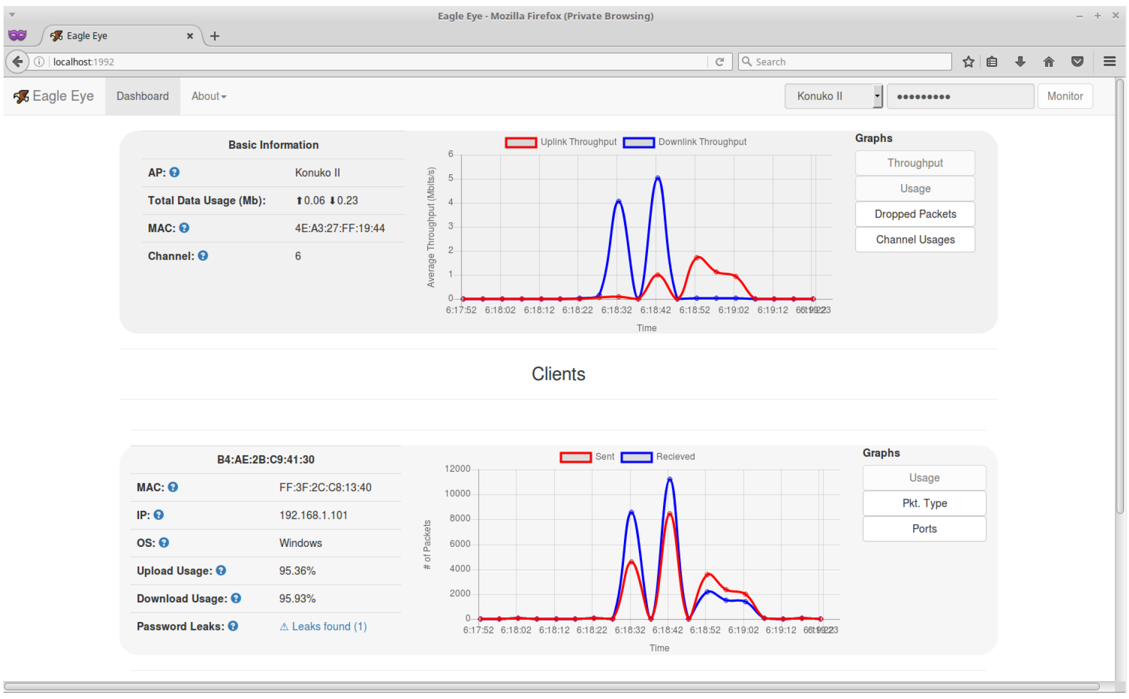
4.1 Packet Capture Accuracy

It was important to make sure the data capture is done in the most accurate manner possible. To test this we set up a small home network using a TP-Link router and 2 client machines. We started an Eagle Eye and Wireshark capture on two separate machines. Then we ran the network for 15 minutes, doing normal web browsing, video streaming, and music streaming on the 2 client machines. Afterwards we saved the internal data-structure produced by EagleEye and the capture file produced by Wireshark. Finally, since EagleEye can also run off a capture file (.pcap), we simply fed the capture file to our monitor to compare the 2 data-structures produced at the end. We ran this experiment a multiple times, and found that the data-structures created were identical in all cases. This means that EagleEye is not dropping any packets, and the information captured is accurate.

Moreover, we compared the information displayed by the monitor, such as 'Dropped Packets' and compared it to what Wireshark reported. This allowed us to validate that our calculations and measurements were done correctly.

4.2 Front-End Responsiveness

Another important characteristic, was for our tool to be responsive and for the usage experience to be as clean an quick as possible. To test for this simply tested the time it took from the time a packet was received to the time it was accounted for and shown on the user's front-end. Now, it is worth noting that our front-end refreshes every 5 seconds (which is equivalent to the size of our time-steps). Thus, we would expect that for any given packet we would want that packet to be shown in the front-end in at most 5 seconds. To test this we simply modified the front and back-end to use instantaneous time windows (meaning it would refresh on every single packet received. Then we crafted a capture file that contained 100 packets and we fed that to the EagleEye monitor. On our front-end we made sure to log the difference between the packet's sent time and the moment it reached our client. Using a standard i5 machine running Mozilla Firefox on Linux Mint 17, it took on average 74.2 milliseconds (with a 2.33 std. deviation) for packets to traverse our entire program flow from capture to front-end. Thus, we proved that our tool is highly responsive.



Basic Information	
AP:	Konuko II
Total Data Usage (Mb):	⬆0.06 ⬇0.23
MAC:	4E:A3:27:FF:19:44
Channel:	6

B4:AE:2B:C9:41:30	
MAC:	FF:3F:2C:C8:13:40
IP:	192.168.1.101
OS:	Windows
Upload Usage:	95.36%
Download Usage:	95.93%
Password Leaks:	⚠ Leaks found (1)

Figure 3: Screenshot of the Eagle Eye Client Application.

5 Limitations

While there are still many features that could be added to EagleEye, we believe that the main limitations that should be addressed are the following:

WPA/WPA2 handshakes One of the main issues that we encountered when developing EagleEye, is that to be able to properly decrypt WPA/WPA2 packets, you not only need the password, but also the initial client handshake. This means that if EagleEye won't be able to properly monitor clients that were already running previous to EagleEye's execution. This issue is not currently addressed, as we assume most network administrators will be able to reset their computer's connection after running our monitoring software.

Lack of 5GHz support Currently, EagleEye only works on the 2.4GHz spectrum, which is arguably the most common spectrum used by most networking devices. However, as devices using the 5GHz band become more accessible, this feature will become a necessity.

Measuring usefulness EagleEye's current design is based on our initial interactions with users, and what we believe would be helpful for them. However, for this tool to be as useful as possible, it is important to continue engaging with the users to fully gauge the usefulness of further modifications or additions.

6 Future Work

The idea behind EagleEye was to highlight the importance of good networking tools that empower non-technical network managers, and to start defining a concrete implementation of this. We believe that for this tool to reach it's true potential there has to be further research and continuous engagement with these users to find the best additions for the tool. We also believe that the tool itself should contain guides and more expansive help sections that quickly teach the user the basics of networking, thus closing the user's knowledge gap.

Aside from the limitations presented in Section 5, we believe there is one final addition that could make this project more complete, Automated notifications. Having a notification system could save users time while trying to catch rare or otherwise transient misconfiguration or errors. It could also be extremely useful for security reasons, as it could alert users of machine activity at irregular hours, leaked passwords, or other suspicious behavior.

7 Conclusion

There is still a lot of work to be done to provide non-technical users with proper tools to effectively manage and troubleshoot their personal networks. As the Internet becomes more pervasive and the IoT trends takes flight, the need to bridge the networking knowledge gap becomes more imperative. We believe EagleEye is a step in the right direction of empowering users to properly manage their own networks. Lastly, to encourage future development, we are releasing EagleEye as open source software, which can be found at https://github.com/pmsosa/EagleEye.

References

[1] Parks Associates, "Streaming media devices: Trends and innovations," Sept 2016.

[2] BBC News, "'smart' home devices used as weapons in website attack." http://www.bbc.com/news/technology-37738823, September 2016. [Online; posted 22-October-2016].

[3] S. Sundaresan, N. Feamster, and R. Teixeira, "Measuring the performance of user traffic in home wireless networks," in *International Conference on Passive and Active Network Measurement*, pp. 305–317, Springer, 2015.

[4] K. Papagiannaki, M. D. Yarvis, and W. S. Conner, "Experimental characterization of home wireless networks and design implications," in *Infocom*, 2006.

[5] A. Patro, S. Govindan, and S. Banerjee, "Observing home wireless experience through wifi aps," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, pp. 339–350, ACM, 2013.

[6] G. Combs, "Wireshark." https://www.wireshark.org, 2016.

[7] Datadog, Inc., "Datadog." https://www.datadoghq.com, 2016.

[8] Paessler, Inc., "PRTG." https://www.paessler.com/prtg, 2016.

[9] Logic Monitor, Inc., "Logic Monitor." https://www.logicmonitor.com, 2016.

[10] Spiceworks, Inc., "Spiceworks." https://www.spiceworks.com, 2016.

[11] M. Uma and G. Padmavathi, "An efficient network traffic monitoring for wireless networks," *International Journal of Computer Applications*, vol. 53, no. 9, 2012.

[12] C. So-In, "A survey of network traffic monitoring and analysis tools," *Cse 576m computer system analysis project, Washington University in St. Louis*, 2009.

[13] J. H. Radek Krejcí, "Overview of local network monitoring tools," *CESNET Technical Report*, 2015.

[14] M. Fontanini, "dot11decrypt." https://github.com/mfontanini/dot11decrypt, 2016.

[15] M. Fontanini, "libtins." https://github.com/mfontanini/libtins, 2016.

[16] SecDev, "Scapy." https://github.com/secdev/scapy, 2016.