Application-level Network Behavior Analysis and Anomaly Detection using Density Based Clustering

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Michael Andreas Anastasakis

Cyber Security Research Group
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Thesis Committee:

Chair: Dr.ir. J.C.A. van der Lubbe, Faculty EEMCS Cyber Security, TU Delft
University supervisor: Dr. Christian Doerr, Faculty EEMCS Cyber Security, TU Delft
Committee Member: Dr. ir. F.A. Kuipers, Faculty EEMCS Cyber Security, TU Delft

Contents

List of Figures

Chapter 1

Abstract

Nowadays, organization networks are facing an increased number of different attacks and existing intrusion and anomaly detection systems fail to keep up. By focusing on security policies, malicious signatures or generic network characteristics, existing systems are not able to cover the full landscape of attacks. In this thesis we try to tackle the problem of anomaly detection on a user network behavior level and an application level. In the proposed framework, network traffic is first separated into different flows based on the mobile application it originates from. Moving forward, the processed network flows are used as input for a flexible noise tolerant behavior modeling framework. The proposed framework is based on density based clustering and tries to identify temporal changes in the user behavior that qualify as anomalous. Moreover, we utilize the model to identify behavioral patterns shared by users and analyze the temporal consistency of user network behavior. To evaluate the framework performance, real network mobile traffic provided a private organization is used. The framework validation is performed by combining the captured network traffic with a conducted employee survey. Overall, the system is able to accurately follow changes in the user behavior based on each application, identify anomalies as well as provide insight on shared behaviors or reoccurring behavioral patterns.

Chapter 2

Introduction

The number of mobile users, that being laptop, tablet or smarthphone has increased tremendously in the past years alongside with the number of mobile applications provided. According to Comscore [1], already in 2015 the number of mobile-only internet users exceeded the number of desktop-only internet users. Based on the same research, mobile represents 65 percent of all digital media time, with mobile apps being the main method of usage. Nowadays, the functions of mobile devices have expanded from simple social communications to various tasks in our everyday life from online purchases to bank transactions. This increase in mobile functionality as well as portability has led businesses to pursue models that take advantage of these features with the goal of improving their efficiency

2.1 Bring your Own Device (BYOD) model

The concept of Bring your own device (BYOD) is a phenomenon that has taken over the business world and has been adapted by many companies. BYOD is a concept which allows employees to bring and use their personal devices in the office, connect them to the company network and conduct business operations. Employees are able to access the company data and services through the devices that they are familiar with and use in their every day life. These devices can range from personal laptops, tablets to mobile devices. A survey by SANS Institute in 2012 argued that 60% of companies already permit BYOD[2]. The reason behind this trend is that BYOD increases the employee efficiency, mobility and satisfaction since they feel more comfortable and familiar working on their own devices[3]. From an economical aspect, the models reduces the expenses of the company as well. The BYOD market size is estimated to be valued at USD 350 Billion by 2022, as per a new research report by Global Market Insights [4]. This provides the proper financial motive for more companies to follow the BYOD trend. Moreover, a similar model that falls into the same category is Choose your Own Device (CYOWD) where companies provide the employees a range of mobile devices to choose from and freely use for both personal and company related purposes. The two models exhibit the same characteristics regarding cyber security and therefore are into the scope of the research.

2.2 Security Threat Landscape in BYOD

As beneficial as these models may be, the introduction of personal devices to the internal company network raises security issues that need to be addressed. Since employees use the devices for activities that are not work related, it leads to the co-existence of personal and company data. The company data accessed by the employee device may contain credentials, confidential emails and documents or company client private information. Leakage, tampering or corruption of such information can have devastating results for the company as well as the individual. The exposure of personal or client sensitive information stored can cause financial and reputation damage to the company. This can happen in the case the employee leaves his device unattended, connects it to an unsecure open Wi-Fi network or installs unverified applications. To get a better insight on the landscape of threats in such an environment we divide them into two categories technical and human related ones that we discuss below.

2.2.1 Technical Threats

People usually install applications on their device without fully understanding the permission requested by apps or the validity of the installed applications. This can lead to situations where malicious applications such as malware are installed on the device. When it comes to mobile security, malware is one of the main attack vectors. The methods of malware infection can be as easy as a user visiting an infected website or downloading a seemingly harmless application. In fact, various technical reports [5, 6] have discovered numerous applications on Google Play store with totals of 100,000 installations that are infected by malware. A mobile threat report by Intel Security Mcafee in 2016 [5] indicates that there has been a 72% increase in the collected unique malware in their labs. This mobile malware growth is taking advantage of the mobile application growth as well as the lack of cyber security awareness end users have. Based on the Mobile Security and Risk Review by MobileIron [7] a known malware, Hummingbad, was able to infect nearly 85 million devices and possibly extract personal or company related information. Similarly, an exploit that does not require the user to actively download and install an application but rather relies on the underlying kernel libraries of the Android OS is Stagefright. The daily detection of unique infected devices from Stagefright is around 800 [5] which means that although new security measures and updates are applied there is still a number of users harmed. Malware not only grows in number but also continues to evolve to bypass existing antivirus, firewalls or other IT related security mechanisms. Moreover, the methods in which malware is distributed, sold or even advertised have advanced and attackers can more easily obtain and perform malware related attacks.

When discussing about such threats not only the private sectors needs to come in mind. The same cyber security concerns are raised on the public one. As technology advances and different devices, applications are used to conduct vital processes in public organizations, the cyber security aspect needs to be taken into consideration. Based on a report by MobileIron [7] the healthcare organizations with at least one compromised device accessing corporate data is at 17% and the government organizations that had a compromised device

accessing corporate data in Q4 is at 9%. One of the most important sectors is the healthcare one since the protection of patient private health information is critical. An overview of the current situation in the healthcare IT system and how BYOD has introduced threats that need to be addressed is presented in [8]. Mobile devices holding and transferring patient records, open networks accessible by patients lead to a situation where a system needs to be in place to ensure the protection of patients' sensitive information. Researchers indicate that 41 percent of people using smartphones in the healthcare sector have no password protection on their devices and 53 percent of them have willingly connected to unsecure of unknown network with their devices. An example that showcases the vulnerabilities introduced in such a environment is the vulnerability assessment performed by a Kaspersky researcher who was able to hack into a hospital and obtain patient records by accessing the medical devices through the hospital WiFi [9]. As a result, we understand that the security protection of organizations that handle vital private information needs to be improved.

With such a broad attack vector on mobile devices and lack of established protection mechanisms, we understand that companies need to be cautious with the devices entering their network. If client or employee information are made public, this can damage the image of the company as well as its business future. Such an impact gives financial motive to rival companies or organizations who want to harm their competition. When it comes to the public sector, a compromised employee device can disrupt critical government services or leak information of utmost importance.

2.2.2 Human related Threats

Apart from the technical aspect of the mobile security threats, another factor that has a high impact is the human one. It is often said that people are the weakest link in the security chain. The security awareness of the everyday user is struggling to keep up with the security risks that arise. Most cyber criminals try to compromise security in various ways such as creating and emulating an unsecured Wi-Fi network in public facilities that being an airport or a hospital. Unaware users will connect to such networks and effectively give free access to their device and personal information that will flow through the network.

Another approach that counts on the human factor is a phishing email attack. A recent publication by Z. Benenson [10] at Blackhat 2016 tries to explain the reason behind phishing attack success. A phishing email experiment followed by a survey of the subjects, revealed that the main reason for clicking on a malicious link inside an email was plain curiosity. An interesting finding of the research is that the IT security knowledge around email spoofing was not statistically correlated with the reported clicking on the survey. Since the nature of this security threat is not entirely technical, the mitigation methods can be challenging.

Furthermore, a human related threat is when the employee willingly acts maliciously or exfiltrates sensitive data out of the company network. In a BYOD environment such an employee could upload to a remote server, send outside the network sensitive data or simply load the data on his personal device and thus cause the same financial harm as in the previous case. The detection of a disgruntled employee can be challenging task from the security aspect. Finally mobile devices are more easily lost than a PC [11] which means that company data if not properly protected on the device can be leaked.

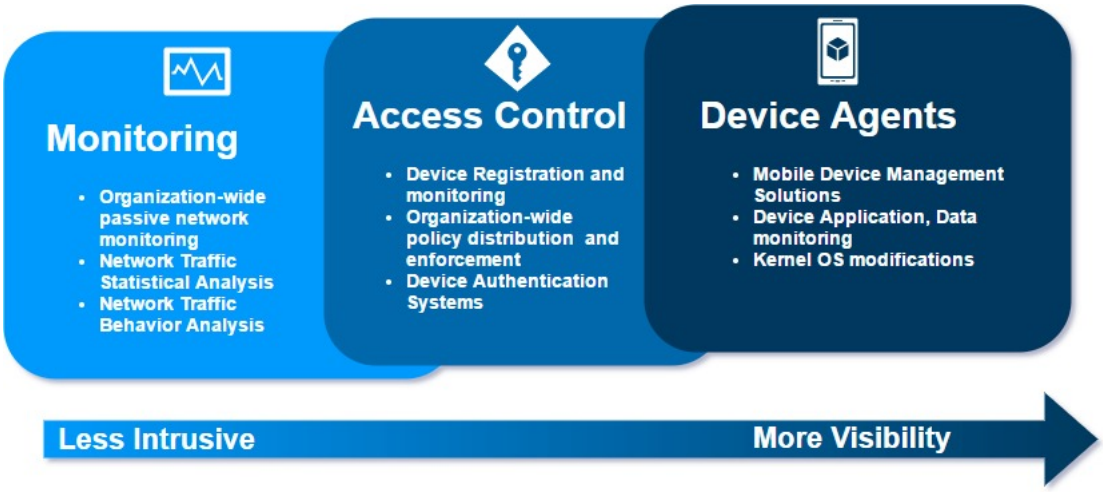

Figure 2.1: Mobile and BYOD Security Management

2.3 BYOD Security Approaches

For organizations to harvest the benefits of BYOD but at the same time secure themselves against the aforementioned cyber security threats, existing solutions try to tackle the problem from different angles. The range of solutions that organizations have to mitigate these threats with can be visualized in Figure 2.1, moving from the least intrusive method that revolves around Network Monitoring and Access Control up to the more intrusive but at the same more fine grained approach of Device Agents. To better understand these solutions and which one of them is followed in this thesis, examples of current research from the network aspect which includes Monitoring and Access Control and the device perspective which includes the Device Agents are going to be presented [12].

2.3.1 Device Agents

The personal devices containing company data can be considered the entry point of an attacker into the company network or the weak point in the data exfiltration process. In order to protect these devices, enterprise solutions introduce the concept of Mobile Device Management (MDM). Usually this type of solutions such as MobileIron have a server, device agent architecture. The agent on the device is effectively monitoring all the device actions and returns information back to the server. From that point on, based on the company security policies the server can enforce actions such as data wipe, device encryption or device isolation. Rhee et al [13] describe such a MDM framework for companies that want to utilize the BYOD concept. Their architecture consists of an agent application installed on the employee device and a device management server. The server is responsible for policy management, enforcing decisions in case of violations while the agent application monitors the device, permits the installation of authorized only applications and sends audit reports periodically to the server.

A similar approach is introduced by Lee et al. [14] in order to handle malicious applications on employee devices. Based on their suggested framework, the installed agent has an embedded White-List that contains authorized applications. The agent checks the device configurations and applications and compares them against the White-List. In case an application is missing from the list, the violation is sent to an administrator to take corresponding actions. Armando et al [15] approach the problem on an application market level. The authors have implemented a proof of concept application named BYODroid. Their architecture consists of a BYODroid market and BYODroid installer on the employee device. Applications listed on the market have a security score based on a source code review. The BYOdroid installer verifies the applications installed on the employee device based on their score from the market and either forbids or permits the device registration to the corporate network.

Apart from the MDM schema, the more technical approach of kernel isolation or virtualization is suggested by various researchers. Such frameworks require the modification of mobile operating system in order to ensure the protection of company information. Kodeswaran et al. [16] propose a framework for run-time enforcement of access control policies in order to protect corporate data. In order to differentiate corporate from personal data, they manually classify mobile applications as corporate and therefore data related to them are also classified as corporate either statically or based on data flow. The architecture consists of an on device policy manager that stores security company policies describing accepted application and data related actions. When an application requests data, information regarding the application, the type of requested data are forwarded to the policy manager who either permits or prohibits the action. A more low level approach is proposed by Oluwatimi *et al.* [17]. The authors made a modification to the Android OS and used a data shadowing approach combined with the Android permission mechanism to effectively hide corporate data from the rest of the user installed applications.

By describing various research approaches on the MDM section, various disadvantages come to the surface. As we can see, most of the MDM solutions focus on the installed applications on the device and whether they follow specific company policies that are related to these applications. The network structure of the company as well as access to network entities that under circumstances should not be accessed by devices are out of the MDM solution scope. Moreover, all of them require from the employees to install a company software on their personal devices that will effectively monitor them. This invasion of privacy can cause a frustration on the employee side which is against the goal of the BYOD philosophy. Moreover, some of these mechanisms rely on a reputation system, known malware history that can be easily avoided by techniques such as polymorphic malware. Furthermore, the approaches that require kernel modifications are not practical on a BYOD environment. Employee may not be comfortable with altering their personal device OS since that can have technical implications on the device. Finally, from a more practical point of view, the MDM approach is not viable for visitors or other who require temporary access to the organization network and therefore such individuals are not properly monitored.

2.3.2 Monitoring and Access Control

When the BYOD concept is used in an organization, all the devices are connected to the internal company network. This means that information both personal and company related flow through the network before reaching the Internet and this presents a point of control for the organization. Administrators are able to analyze the network traffic in order to identify specific actions or patterns that do not comply with company enforced policies. Based on company policies, it is usually the case that specific servers or devices must be accessed only by a handful of employees. These network actions are into the scope of the Network Level approaches we are going to analyze. Moreover, this creates the opportunity to utilize a big amount of user generated network data. This pool of information combined with the upcoming solutions provided by machine learning approaches, leads to intelligent network behavior and anomaly detection solutions.

Access Control: Every organization has a specific network structure with nodes and information which should not be accessible by everyone. For this reason, security policies are created that dictate the permitted actions based on user roles and positions.

Chung et al. [18] propose a 2-tier framework to enforce this access control on a BYOD environment. Their approach can be considered as a combination of a device agent solution on a small part alongside a network controlling server. The architecture consists of two tiers, a device control tier and a cloud control one. Every employee device has installed a lightweight system that contains an anti malware software as well as a device profile. The profile contains a combination of contextual rules that define which actions are allowed for the device e.g. at what time can this device connect to the company network, which data can this device access. The management system of their architecture is on the cloud and consists of a few elements. There is a profile management system that controls, updates and creates the device profiles as well as a access control log that contains information about the employee actions.

Costantino et al [19] suggest a framework that implements a role-based access control as well an authentication schema based on the employee context and role in the company. Their architecture consists of a policy server and an authentication server. The policy server contains and updates the company policies that dectate the employee actions based on their role. The authentication server is based on the OAuth 2.0 protocol and is responsible for verifying the credentials of every device that is trying to access the company network. When a device connects to the internal network, the authentication server verifies the identity of the device. If the authentication fails, the device is not granted access. As a next step, the policy server validates the device compliance with company policies and permits or denies access to the network.

Finally, a Access Control framework that is closer to the Monitoring aspect is presented by Koh et al [20]. The authors propose a dynamic access control framework that takes into consideration the contextual information of users and creates normal and abnormal behavior patterns. When a user tries to connect to the network, device context information are retrieved and forwarded to a detection system. These information are compared with stored normal behavior information of the same user and the system decides whether the new provided information diverge from the normal ones on such a scale that they can be

considered abnormal. Based on the decision, a policy server enforces the access control policies to the device.

The main flaw of an access control policy system lies in the manual character of company security policies. Most of the time, such rules are created by an administrator who is also responsible for updating them. The introduction of the human factor in this process can introduce errors such as misconfigured policies or outdated ones. Moreover, despite the fact that the authors in [18] start to diverge from the Device Agent approach, their approach introduces the usage of a cloud service which can be a problem for some organizations. When it comes to security, many companies are not fully comfortable with uploading sensitive company information or policies on the cloud. In addition, the concept of the [20] is promising but there are no details related to the normal and abnormal behavior classification as well as the nature of the contextual information they retrieve. Both of these elements are key pieces to a successful dynamic access control framework. Finally, access control frameworks can enhance the security of an organization regarding the introduction of a device to the company network. Most of these approach, fail to properly monitor user behavior afterwards and identify abnormal patterns.

Monitoring: The monitoring approach tries to utilize the network information from an analytic perspective and come to conclusions that will provide insight on the threats in the network. The observation of the traffic behavior and especially traffic generated by the devices in a BYOD environment can provide insight on the type of protocols, applications as well as behavioral patterns. These findings can lead to elements that stand out in the network that can be interpreted as malware or threats towards the network in general. In the industry, security information and event management (SIEM) tools such as IBM Qradar [21] and Splunk [22] are mostly used for this type of passive monitoring. These frameworks monitor network events created by connected devices , analyze packet data as well as monitor user behavior.

Although, there has not been a great amount of scientific research specifically on the network analysis of a BYOD environment, there have been a few contributions. Zeng et al [23] implemented a framework that provides insight on the network behavior that includes traffic behavior, end-to-end behavior, routing behavior and application behavior. The proposed architecture consists of a measurement platform which includes probes to collect the traffic, a control platform that is responsible for displaying the analysis output and contacting an analysis platform which creates the results. After evaluating their approach on a campus network, they were able to efficiently visualize a large scale network behavior and provide insight on the used protocols, services as well as applications in the network.

NetFlow [24] is a protocol designed by Cisco to efficiently monitor network traffic and provide insight on it. Hou et al [25] used NetFlow to create an IP network behavior analysis system. Their system consists of two behavior analysis modules, the network traffic and the user one. After properly formatting the network data and aggregated them into network flows the authors utilize analytic algorithms such as TOPN or comparative analysis to extract information about the protocols, services and ports used in the network. In order to perform the user behavior analysis, information such as visited URLS, type of website visited are used. This type of information is obtained through a deep packet inspection

(DPI) approach. As a result, their system is able to visualize a detailed traffic analysis, a HTTP traffic trend analysis as well as visited website trends. As future work, the authors suggest the automatic classification of traffic based on the extracted characteristics of their system.

A problem with passive monitoring approaches is the interpretation of the generated insight. The information that derive from a network analysis tool can be translated into meaningful information only be people who have the appropriate security knowledge. To deal with this situation, solutions have introduced the machine learning approach that is able to learn from the network data and automatically distinguish normal from abnormal behavior. However, as with every machine learning approach these solutions require training with proper configuration to avoid the common mistake of false positives.

2.3.3 Research Approach

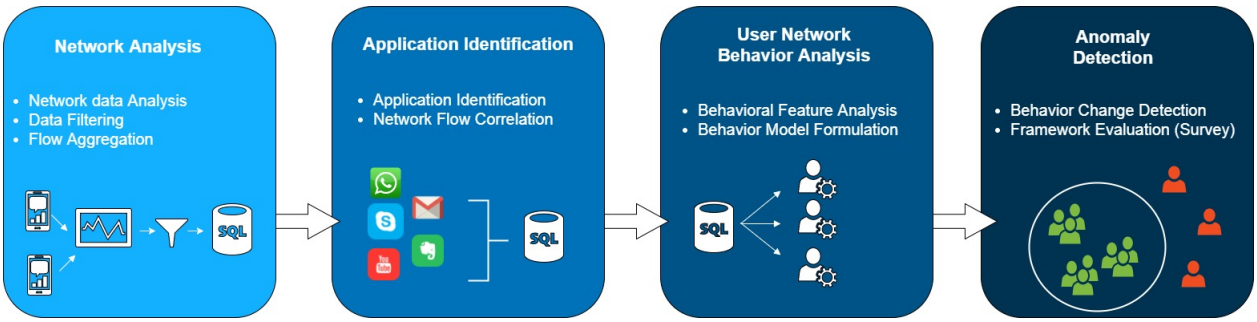

Figure 2.2: Thesis Research Overview

Every solution that is introduced to tackle the BYOD security threats in an organization has specific advantages and disadvantages. Most of the existing research has heavily focused on Device Agent and Access Policy approaches while there is room for development on the network aspect as well as a lack of implemented framework that can validate or contradict proposed models. Moreover, the passive networking approach is the least intrusive compared to the rest which is an important factor for the success of a security model in a BYOD environment. Furthermore, the nature of a network security approach gives the ability to be applied in different scenarios because of its modularity that being a corporate network, a hospital or a government facility. For all these reasons, the focus of the thesis is on network monitoring and traffic analysis.

The research problem is the network analysis of user behavior in a BYOD environment based on mobile application generated traffic. In order to tackle the formulation of a user network behavior model, a framework that performs a cluster flow temporal analysis is created. This is achieved by utilizing density based machine learning algorithms as described later in detail. Regarding the mobile application level aspect, a better understanding of the user behavior can be achieved since an efficient separation of network traffic is applied by utilizing a proposed novel application identification algorithm. The proposed framework goal is to efficiently follow the human behavior reflected on the network traffic and detect

changes and anomalies. This is accomplished by techniques of anomaly detection closely related to the concept of cluster algorithms. A general structure of the steps followed in the thesis is presented in Figure 2.2.

Overall, our study can be formulated into the following research questions and sub-research questions:

1. **Are we able to transform raw encrypted network traffic into sessions separated based on the application they belong to?**
   a) What are the network session fields that provide information regarding the mobile application?
   b) How well can the model identify the application depending on the used protocol?
   c) Is the proposed method able to scale on a large network environment?
2. **Can we implement a framework that translates user network behavior into a measurable metric?**
   a) What are network features that can efficiently characterize a user network behavior?
   b) What is the most suitable clustering algorithm for such a model?
   c) To what extent can such user models be utilized to detect anomalies and behavioral changes in a BYOD network?
   d) To what extent can such user models scale on a large scale organization network?
3. **Can the proposed framework identify behavioral patterns and trends as well as compare user behaviors?**
   a) Is the model able to reflect user behavior patterns on an application level?
   b) Is the model able to detect behavior patterns and application usage habits shared by the users?
   c) To what extend is the model able to identify user behavior stability?
   d) Is the model able to compare user behavior and identify user groups of similar behavior?

In the following chapter, the research approach is broken down into key elements and the related work around them is presented. The presented related work will provide the motivation and detailed reasoning for the choices made in the thesis methodology. Before describing the proposed framework, the choices regarding the network data collection and analysis are presented in Chapter 4. The detailed methodology and design of the framework are explained in Chapter 5. Moreover, the method in which the framework is evaluated is also presented in 5. Finally, the results of the research are discussed in Chapter 6 as well as the conclusion of our research in Chapter 7.

Chapter 3

Related Work

Research in the sector of network anomaly detection has primarily focused on methods that examine network characteristics without taking into consideration the human behavior aspect and the application context of the network flows. For these reasons, the structure of the related work is broken down to the three key elements that formulate the research of the thesis. Firstly the research methods around the application traffic identification in which a vast amount of network flows are attributed to specific mobile or computer applications is analyzed. In this way, the possible methods in which traffic will be filtered and analyzed in this thesis are presented. Furthermore, the broad spectrum of methodologies in network anomaly detection is analyzed. The benefits and limitations of the established methods are presented in order to identify the ones that will contribute to the goal of the thesis. With the two previous aspects covered, the work that is closer to user behavior analysis on a network level is presented. Finally, an overview of the limitations of related work and how this thesis is trying to overcome them is stated.

3.1 Application Traffic Identification

Since the network traffic generated by applications is the aspect in which data will be analyzed, a better understanding on the network behavior and identification of mobile and computer applications is required. Unlike traditional applications and protocols, modern applications do not use specific protocols and IP addresses and therefore can be hard to identify on the network. When it comes to application identification, the research is mainly split in two approaches. The first one is a payload-based method which identifies traffic by checking the existence of specific information on the header or the payload of each packet. Although such approaches have a high accuracy rate, they encounter difficulties related to encrypted traffic, computation complexity and user privacy issues. The second approach is a statistical-based method that analyzes the statistical characteristics of a network flow e.g. distribution of packet size, connection duration, source and destination address. Such methods try to define characteristics that are unique, to an acceptable percent, per application. Similarly as the previous one, this approach has limitations since an exhaustive statistical analysis of all mobile applications is not feasible as well as the misidentification

of applications that share online resources as such CDN servers can create problems.

A very interesting work on application traffic identification is presented by Q. Xu et al [26]. The authors propose a system called FLOWR that is able to identify applications by continuously gaining information about their unique network features through traffic analysis. FLOWR is collecting information from the HTTP headers of application packets and tries to identify which information can uniquely be related to an application. The information can range from specific strings in the URL of an HTTP GET request to strings in the Referer field of an HTTP message. Information that can be generally related to an application are called application identifiers and when they are unique enough to be tied to an application they are called application signatures. An application signature can be for example a unique string in the HTTP header that contains the application Play Store id number. After the authors manually provide an initial seed of applications and application signatures to the system they apply a temporal correlation approach to generate more signatures. When a application identifier re-occurs with a high frequency closely in time to an application signature then it has a high probability of becoming an signature for that application. The authors call this method flow regression and is the main mechanism of creating new signatures. In order to maximize the algorithm efficiency the authors tune the value of the window in which an identifier is considered close in time to a signature as well as the number of co-occurrences that are required to promote an identifier to a signature. FLOWR is able to narrow down 65% of the network flows to 5 or fewer candidate applications as well as accurately associate 86-95% of flows to their generated apps. As the authors mentioned however the algorithm has a few limitations. Most importantly, the restriction to HTTP only traffics can leave out of the scope application traffic that is fully encrypted. Moreover, the initial seed of applications plays an important role on the growth rate and success of the application signatures and limits the number of applications that can be identified. Despite the limitations, this work provides great insight on a non statistical approach to relate network flows to applications.

A similar research that takes advantage of information on the HTTP header is presented by Dai et al [27]. In their research the authors create a novel system that automatically generates network profiles for android applications. The network profile consists of the network flows sent by the application during its execution. In order to profile an android app, the authors have created a fuzzing framework that consists of an android emulator and a fuzzing mechanism. Every application that is under analysis is installed on the emulator and the fuzzing mechanism implemented with the tool monkeyrunner [28] is applied. The fuzzing mechanism generates a sequence of random UI related actions that will trigger the application to create network traffic. The authors have implemented two execution modes a truly random mode and a directed mode that allows a user to communicate with the tested app and insert specific actions when they are required. Moreover, a small analysis is conducted on the ad related network traffic generated by applications since such flows are rich in unique strings in their HTTP headers. After analyzing the HTTP content of each type of flow and the possibility of finding unique identifiers, the author implemented an network fingerprint extraction mechanism called Fingerprint Extractor. This mechanism takes as input the HTTP header, URL and fields strings and creates a tokenized flow after splitting the string in segments. In order to achieve higher accuracy and correlate flows that

should fall into the same application category even though they have minor differences in their string values the authors apply a clustering algorithm. Tokens from the tokenized flow are grouped into clustered based on their similarity scored computed with the Jacard index. Some limitations of the authors work are the inability to distinguish apps which use the same online services and the need for a directed fuzzing mechanism when the application has a log in functionality. Despite the limitations, the authors contribute some important information on the various network flows that can be generated from an android fuzzing process and more specifically the network fingerprints of ad-related flows.

Zuquete et al [29] present a decentralized architecture that identifies the application source of network traffic by using a device agent that modifies outgoing application network packets. The architecture consists of a mechanism on the host device as well as a central network monitor system controlled by an auditor who is responsible for detecting anomalies. Each application on the host is assigned a unique application identifier and a local database is created contained information related to all the installed applications. The database, with the application identifier being the database key, stores the name of the package containing the application, the version of the package and the applications binary local path. When network traffic is generated by an application its corresponding identifier is embedded to each packet and details about the applications are updated locally on the host. From each packet that arrives to the network monitor system, the application identifier is extracted and each host database is queried for a detailed set of application information. Therefore, each flow of network traffic is correlated to a specific mobile application and the auditor has an overview of the application network traffic. However the suggest approach has a few practical limitations such as communication problems with the devices as well as the on device monitoring mechanism efficiency that the author do not take into consideration.

Kumano et al [30] propose a method of multi-stage application identification by using statistical information from the first packets of a network flow. The purpose of the multi stage classification is to first identify the category in which the application falls so that the following classifier can be focused on applications of a specific category. The network features used to create the classification models are derived from network flows belonging to various application categories such as Youtube, Hulu as streaming applications and BitTorrent as P2P ones. In order to narrow down the network features of the dataset that are most suitable for separating network flows the authors focus on the level of variance they exhibit and how uniquely they can be related to specific applications. After determining the appropriate features, the filtered data are fed into a multi-stage classifier that progressively tries to determine the type and category of application. An interesting configuration by the authors is that each classifier uses network features based on a different number of packets and is therefore better tailored to classify specific categories of applications. The concept behind this configuration is that the network traffic generated by each type of applications does not require the same number of packets so as to be uniquely identified. For example, an HTTP web flow only needs a few initial packets to identify the application type since the rest of the traffic will be the actual content of the web page. Based on their implementation, the authors can identify applications from their training set with an 88% accuracy although the method has not been verified against encrypted network flows.

Another research in the statistical section that focused directly on specific applications and not the category they fall into is presented by Amac et al [31]. The authors suggest that a direct classification of applications is important for the security implications specific apps present and believe that a statistical machine learning approach is the most promising one. Their architecture follows a typical supervised learning method which includes a labeled dataset containing network flows from various applications, a feature extractor and a classifier that is fed the features as training data. For the research, the UNB ISCX Network Traffic dataset was used as well as a smaller internally made that included the most recent and widely used social applications. After trying various machine learning algorithms as well as different combinations of network features, the authors were able to achieve an average 87% accuracy on all network flows. However, they observed the phenomenon where some applications had poor performance and were misclassified as other applications serving the same purpose. The main reason behind it is that applications such as WhatsApp and Skype have very similar network flows since they have a very similar behavior and therefore increase the number of misclassifications.

A hybrid approach to identify application traffic is introduced by Y. Won et al [32]. The authors propose a model that has two steps into the identification, a signature matching and a session behavior mapping. The suggested signatures are patterns of hexadecimal digits or strings that can be found in the payload of a packet. A signature is divided into two categories, a packet signature that includes the port number, protocol and specific strings in the header and a behavior signature that includes connection information such as number of protocols and packet burst period. When a new packet is analyzed, its signatures are extracted and compared to signatures from known applications. If there is a match, the packet is allocated to that application and the signature baseline is updated. If there is not an application that matches with a high probability then the session behavior mapping part of the model is enabled. In this case, packets are correlated to previously known processed packets if they share the same origin IP address and port number. The network traffic used to test the hybrid model was obtained by the POSTTECH university network in the timespan of 1 hour. Apart from the typical validation approaches in these cases, where labeled data are used as unknown to calculate the accuracy of the approach the authors introduce a port related angle. Once all traffic was assigned to specific applications, the port dependency ratio (PDR) was calculated in order to observe specific patterns of port usage by applications. The results indeed verify that a number of applications are designed to allocate ports that are close in between them as well as specific ports based on application specific protocols. Of course, there are cases where the port distribution is too noisy to provide meaningful insight. Such an approach however can indicate whether a specific applications uses a fixed port for its network traffic and therefore verify or not the application identification results.

From the aforementioned papers, it is shown that there is a wide range of tailored solutions toward application network identification that rely on either content analysis or statistical analysis. When it comes to content analysis either payload or packet header, the lack of implementations that take into consideration encrypted traffic can be a problem. A considerable amount of traffic generated by applications especially the control packets that establish the communication of the application with a server are encrypted. Apart from that, the ideas presented and some details in their frameworks will contribute to the application

identification approach presented in this thesis. Regarding the statistical analysis ideas, the main obstacle is the lack of a scaling implementation. The need for prior analysis of application in order to identify them later is a great problem for a large organization. Moreover, the requirement of labeled data narrows dows the evaluation of such methods to systems tailored by the researchers. Despite these problems, the analytic related papers provide great insight on the network behavior of applications that will contribute to the comprehension of results from the identification approach implemented in the later stages of the thesis.

3.2 Network Intrusion and Anomaly Detection

Intrusion and prevention detection systems are key elements of the cyber security of an organization network. Such systems are configured to monitor the network in order to identify potential cyber threats or risks based on various methods. Since the goal of the thesis is to implement a system regarding network anomalies and before diving into the user behavior anomalies, the existing methodologies and knowledge around anomaly detection needs to be established.

Although there have been various research proposals on anomaly and intrusion detection systems, all the proposed ideas are based on four main methodologies. An overview of these four main methods is presented by Mudzingwa et al [33]. As displayed in Figure 3.1, the methodologies can be broken down to three distinct ones and a final method which is a combination of the rest. All of them share the same architecture and differentiate only on the method of information analysis and determining whether there is a violation. The architecture was developed by the IDWG [34] (Intrusion Detection Working Group), has four functional modules as show in Figure 3.2 and consists of the following blocks:

- *E Blocks (Event)*: This type of blocks are responsible for monitoring and collecting the information from the target network.
- *D blocks (Database)*: Blocks that are used to store the information from E blocks for further analysis by A and R blocks.
- *A Blocks (Analysis)*: Processing modules that analyze the provided information in order to determine whether there is an anomaly.
- *R Blocks (Response)*: Blocks that are responsible for responding to intrusions and block then.

The Signature based methodology seeks for defined patterns, or signatures, when processing the analyzed data. These signatures are defined a priori and can be related to attacks ranging from malware to network malicious actions. When a signature is observed in the network that matches the malicious defined ones an alert is raised. Some of the benefits of signature based methods is the fact that they are fast and easy to deploy since there is no model training period as there is for some of the following methods. The system only searches through the dataset and compares to a known signature list. Despite the small overhead and the high accuracy to known attacks, the method has a number of disadvantages.

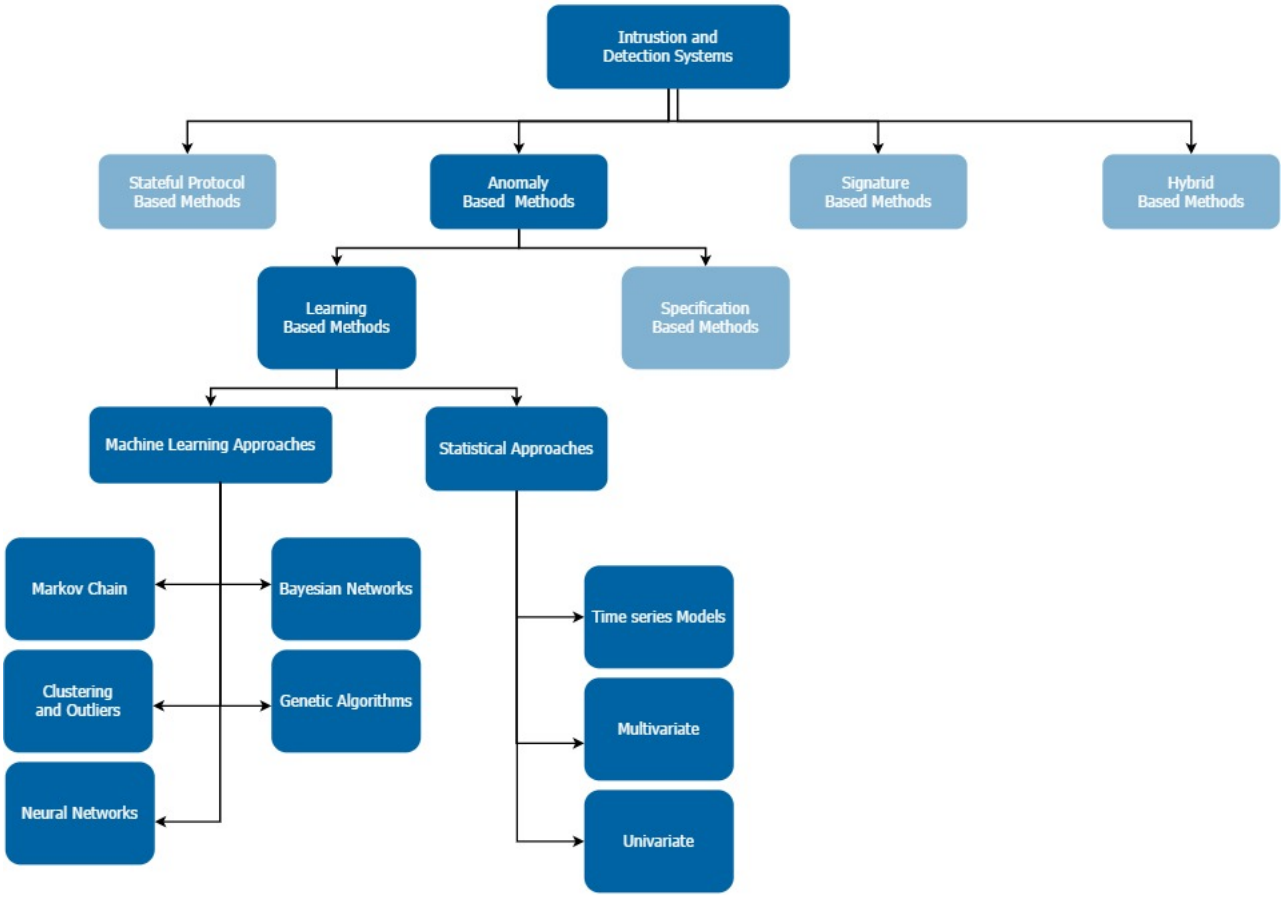

Figure 3.1: Intrusion Detection Overview

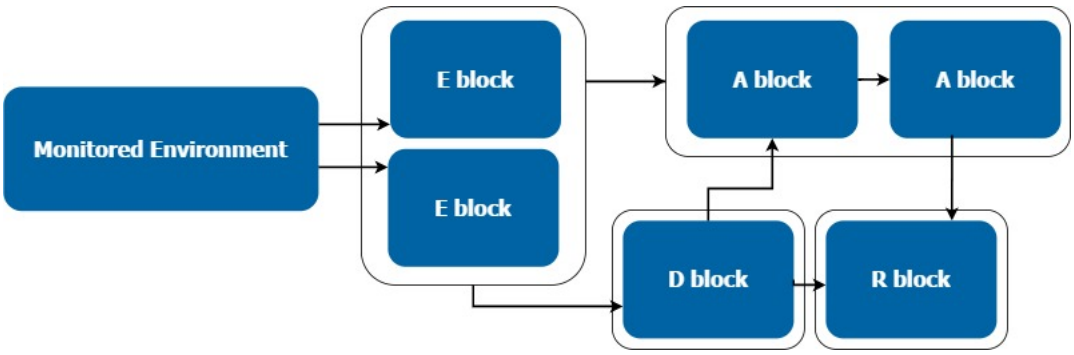

Figure 3.2: Intrusion System Architecture

The comparison of new elements to a pre-defined set of information automatically means that the method is unable to detect new attacks or at least attacks that are not included in the set. This can occur when an attacker modifies a known attack so as to evade the system or the person who is responsible for the intrusion detection set has not updated it.

The Stateful Protocol Analysis based methodology operates by comparing identified protocol profiles against the monitored network behavior. These profiles are designed and determined by vendors and industry leaders as the accepted definition of benign activity. In this category fall the communication protocols of various applications and their normal protocol flow e.g. every request needs to have an a predictable response and those that fall outside of the expected result are flagged. This methodology is similar to the signature based one with the difference that it has a better and deeper understanding of how protocols and applications are supposed to work. However, it introduces some limitations such as the inability to detect an attacker who is able to exploit the system without diverging from the normal protocol behavior. Similarly as before, the requirement of a pre-defined set of protocol profiles can introduce problems for the methodology.

The Anomaly based methodology compares the monitored network behavior against established baseline profiles. Such profiles are created during the learning period where the method trains a model based on normal network behavior of the analyzed environment. The used profile can be either dynamic or static. A static profile is created before the monitoring phase of the system and does not change while the dynamic profile is updating throughout the monitoring phase. This methodology is able to detect zero-day attacks with a high accuracy as well as identify anomalies closer to the human nature and less technical related. However, when a static profile is creating the number of false positives is high since a user can change his normal behavior and misidentified by the system. At the same time, when a dynamic profile is used an attacker can spread the attack over a long period of time and as a result make the attack part of the profile. For these reasons, great attention need to be paid to the used training model and its configuration so as to exploit the advantages of such automated modeling approaches with the minimum number of errors

Because of the static nature of the first two methodologies and the highly dynamic aspect of user network behavior, the anomaly based methodology is the one that is going to be used for the purpose of the thesis. Moreover, Signature based or a Stateful Protocol based approaches are only able to detect anomalies that have been observed before and are only connected to well known attacks. However, anomalies in the human behavior can be unpredictable and very hard to relate to specific network attacks.
There is a number of different anomaly detection algorithms based on the machine learning approach used or the distinct variables configuration. However, all the algorithms fall into some generic categories. A review on the available categories regarding the anomaly based intrusion detection is presented by Jones et al [35]. The authors divide anomaly detection methods into two main categories, learning-based and specification-based ones. The specification-based methods which are protocol based, state-based and transaction-based fall into the general category of approaches which have a static nature. For this reason, they are not part of the analysis. On the other hand, the learning-based methods are based on the application of machine-learning techniques or statistical methods to define a model of normal behavior. Extensive work on anomaly detection and the learning methods has also

been conducted in [36, 37]. A combined overview of the methods that can contribute to the thesis is presented in Fig 3.1.

3.2.1 Statistical based anomaly detection

In statistical-based anomaly detection techniques, the system traffic is captured and user actions are observed in order to create profiles that reflect network behavior. The profiles refer to features such as number of packets, traffic volume, number of connections etc. During the anomaly detection process there are two different profiles that are used for the classification. A stored behavior profile that is trained by using the previous seen network traffic data and the current profile. As network events occur, the current profile is compared with the stored one in order to measure how much the current network activity diverges from the one seen before. The degree of irregularity on the behavior is the metric used to classify it as abnormal if it surpasses a specific threshold.

Statistical approaches have a few advantages when applying intruction detection. An important benefit is the lack of need for prior knowledge about the previous network behavior. The models are able to learn the expected normal behavior through observing the network. Moreover, statistical-based methods have a good accuracy when it comes to attacks occurring over long period of time and are good indicators of denial-of-service (DoS) attacks. Despite the advantages, the methods also present a few flaws in their approach. The most fundamental problem comes with the concept of a statistical approach. User behavior can be rather random and translating it into a normal distribution can be a challenge. However, statistical approaches require a distribution that will be used to compare features and decide on the classification. Therefore, the approach can be sensitive to data that easily change and therefore increase the detection errors. Furthermore, it can be difficult to define the threshold between normal and abnormal behavior and create a balance between false positives and false negatives. Moreover, an attacker is able to evade the system by performing an attack whose features fall into the acceptable levels of normal behavior. Below a few research papers around statistical-based anomaly detection related to the concept of the thesis are presented.

A histogram-based traffic anomaly detection methodology is proposed by Kind et al [38]. The authors construct histograms in order to describe the characteristics of traffic features and gain insight on the distribution of the features. The network featured used are the typical network information extracted from a packet header e.g. source IP, destination IP, port number. For each feature the authors created a set of training histograms containing normal network behavior which are mapped into a metric space. The space was created in such a way that similar histograms have a small distance and dissimilar ones are far away. In order to measure the distance between histograms the authors tried different distance metrics in order to take into consideration the variance and the value range of each histogram. As a next step, a clustering algorithm is applied in order to cluster similar histograms together. Any new incoming traffic will be translated into a vector that will be compared with the existing clusters. if the vector falls within the clusters then the behavior is considered normal. Otherwise, the behavior is considered abnormal and the severity of the anomalous behavior is measured by the distance of the vector from the clusters. In order to evaluate

the framework, a trace of NetFlow packets from the IBM Research campus at Zurich and a trace from a data center with high-volume websites were used. After optimizing the configurations of histograms as well as various clustering parameters, the framework was able to detect 13 out of 15 attacks which included port scanning, worm propagation, mail bombs and system fingerprinintg. However as the authors point out, there is a medium number of false positives with a small distance from the clusters that can be mitigated with an optimization on the distance threshold for raising alarm which they suggest as future work. In general, a histogram-based approach as well as a distance metric similarity of them is an interesting approach on anomaly network detection.

Limthong et al [39] identify anomalous activities from unwanted traffic data through Discrete Wavelet Transform (DWT) techniques. The suggested methodology is to interpret packets into a wavelet function since wavelets achieve good frequency resolution at low frequencies and good time resolution at high frequencies. Therefore, a decomposition of a wavelet that contains abnormal elements will bring them to the surface. Based on the authors, the abnormal activity that needs to be identified on the network consists of specific packets that are translated into known type of attacks. More specifically, the network domain taken into consideration is the traffic in the Darknet and the type of packets are TCP SYN, TCP SYN/ACK and UDP packets. The types of attacks that can be detected through this method include port scanners, backscatter and flooding attacks. After generating the wavelets based on the distribution of the three specified type of packets, the authors perform a wavelet decomposition in order to gain insight on how clearly attacks can be identified. After experimenting with various values for the time interval and wavelet levels the authors are able to identify abnormalities in the wavelet decomposition that can lead to the detection of attacks.

Beach et al [40] propose a framework that detects anomalous behavior through forecasting algorithms on the network packet header information. By using a network dataset from Fermi Lab that spanned one day, the authors were able to exhibit specific cases of malicious network behavior and their unique characteristics. After filtering the dataset into flows based on the IP addresses and ports, they applied the Holt-Winters algorithm in order to predict the future values of network traffic. When a value highly deviated from a defined threshold it was considered as an anomaly. Moreover to increase the accuracy and validity of the results they enforced a sliding window on each examined value and when the number of anomalies around it was above a specific number then the value was indeed considered as an anomaly. On the results section, the authors describe specific cases of abnormal behavior with increasing complexity that need to be taken into consideration for our research as well. They describe cases such a simple one-variable abnormality based on one port activity and the importance of seasonal effects on a network and how they can be misconcepted as abnormalities without proper scaling on the analysis. Moreover, they demonstrate cases where network port actions need to be correlated in order to get meaningful insight on malicious events on the network.

Ding et al [41] try to describe the characteristics of a network and identify abnormal patterns with the usage of a Traffic Matrix (TM). A TM consists of all the flows inside the network between nodes. Each row represents the network activity of a node towards the rest. Since such a representation can have a sparsity problem on a large scale network the

authors apply a principal component analysis (PCA) to analyze effectively the TM. After performing the PCA, the dataset is structured in a way where the first principal component value contains the highest amount of information. In order to define a network flow as anomalous, the authors use two parameters a dissimilarity metric and an anomaly score. The dissimilarity metric of a node is based on the distance between the PCA values on the node from the rest of the set. The anomaly score is defined as the ratio of the projection of data onto the first principal component value. After calculating the scale of these two parameters and understanding the normal range of values, a flow of a node is considered anomalous when one of its two parameters is out of the normal scale. The authors decided to use these two parameters in order to increase the accuracy of their algorithm and decrease the false positives. When compared to other existing anomaly approaches, their suggested framework has a smaller percentage in False Positives and False Negatives. However, the creating of a matrix to represent the network flows of a monitoring environment introduces a scaling problem that decreases the efficiency of the approach.

3.2.2 Machine learning based anomaly detection

In machine learning-based anomaly detection techniques, traffic is collected, properly filtered and provided as input to a machine learning algorithm. In general machine learning algorithms require a training dataset as input that will provide information on what is the expected output values, in this case expected behavior, and will try to optimize their performance with the goal of predicting such behavior. Unlike statistical approaches, machine learning methods adapt and improve their performance based on every new incoming information. This dynamic adaption advantage comes in the price of a resource expensive procedure to train a machine learning algorithm which is also the main disadvantage. In most of the cases, a rich labeled dataset is required for an algorithm to perform efficiently and the acquisition of such a dataset can be an obstacle. Machine learning algorithms can be separated in two main categories, supervised and unsupervised learning. The first case includes all the methods that require a labeled dataset for the training phase. On the other hand, the second case includes algorithms that are able to perform an analysis on an unlabeled dataset. Because of the extensive amount of research on machine learning based anomaly detection in general, it is more useful to provide a general overview of the different algorithms used. Moreover, two selected papers on machine learning anomaly detection that focus on unlabeled data and contribute to the concept of the thesis are presented in detail.

As seen in Figure 3.1 there are 5 main machine learning approaches that are applied on anomaly detection. Each approach is using a different method to tackle machine learning problems as well as a different type of dataset to achieve that.

A Bayesian network is a probabilistic model that represents a set of random variables and their conditional dependencies. Because of the representations of interdependencies between variables, a Bayesian network is useful when part of the data are missing. Moreover, the use of acyclic graphs to create a Bayesian network introduces casual relationships between nodes which can be translated into consequences of specific network actions. In general, Bayesian networks have been used in intrusion detection alongside statistical

schemes. However, based on Kruegel et al [42], they exhibit similar results with statistical approaches while requiring higher computational effort. Moreover, the classification of anomalies is highly dependent on the values that are considered normal on the created probability table.

A Markov Chain is a stochastic model used to model a system with the assumption that future states can be solely defined based on the current state. Markov chains have been highly used in anomaly detection because of their ability to represent event sequences and translate them into a graph of nodes. By analyzing such a dataset of event sequences, a Markov chain model calculates the probability of an action occurring based on the current state. When an action with a low probability based on the model occurs then it is more likely to be considered anomalous. One major drawback of Markov Models is the high increase in computational requirements as the number of network connections between nodes that are going to be included increases as well.

Neural Networks have been used in anomaly and intruction detection because of the adaptivity and flexibility they exhibit to changes.One main advantage of neural networks is their ability to predict a value using imprecise data that include a certain degree of uncertainty. However, most neural networks fall into the category of supervised learning and thus are out of the scope of the thesis.

Genetic algorithms are used to solve optimization and search problems based on the concept of evolutionary computation. Such algorithms are used in anomaly detection mostly to decide on optimal features and parameters that will be further used for the detection process. The efficiency of genetic algorithms on deciding optimal parameters is based on the mutation and crossover steps of the algorithm and has seen success on the network anomaly detection.

The final category of machine learning algorithms are clustering and outliers. As the name suggests, observed data are grouped into clusters based on a similarity or distance metric calculated from their features. In anomaly detection applications, outliers or small clusters can be considered as data that appear with a smaller frequency or have a specific feature value that makes them stand out of the perceived normal behavior. An important advantage of clustering algorithm is the lack of need for labeled data since they focus on the existing observed data and their in-between similarity so as to define what is abnormal.

Portnoy et al [43] present a clustering-based anomaly detection algorithm that uses unlabeled data. The authors decide to investigate the efficiency of unsupervised algorithms since as they state there is an abundance of proposed supervised machine learning detection algorithms. The dataset used for the clustering is derived from the KDD Cup 1999 Data that contained a variety of intructions simulated in a military network environment. From the dataset the authors extracted network features of TCP connections such as duration, protocol type, traffic volume, number of connections. However, they ignore the label provided by the dataset so as to exclude the information whether a connection is part of an attack or not. In order to perform a clustering algorithm, the connections are compared on a space created based on their features and the Euclidian distance is as metric. The motivation behind clustering is the assumption that normal and anomalous traffic form different clusters in space. After applying the single-linkage clustering algorithm, clusters of connections are creating and based on the number of cluster members they are labeled as normal or abnor-

mal ones. When the training and clustering is completed, a test set is provided where each new connection is labeled based on the label of its closest cluster. The authors examine all the possible parameters that can optimize a clustering algorithm as well as the network context information of a dataset and how it can affect the results. On their evaluation, they determine that the percentage of normal and abnormal connections in the training set plays an important role on the final accuracy of the algorithm as well as the balance between detection and false positive rate can be a challenge. Moreover, the authors state that this approach is unable to detect malicious intent of authorized users and focuses on known network attacks. Despite these potential problems, their proposal has promising efficiency and the lack of need for a labeled dataset makes it an attractive solution. The experiments and the analysis on this paper provides great insight on the potential obstacles to consider and how the focus of this thesis can overcome these obstacles and limitations.

A similar research regarding the clustering aspect is presented by Munz et al [44]. The authors apply a flow-based anomaly detection based on the K-means clustering algorithm. The approach is also based on the same assumption that flows on normal and abnormal behavior will form different clusters. In this case, the authors try to detect time intervals in which the network traffic is abnormal. The established steps of filtering and feature extraction are applied on the training data where the features are the total number of packets, total number of bytes and number of different source-destination pairs. An important aspect of the research is the labeling of traffic based on the port used in order to cluster traffic based on specific services e.g Port 80 traffic is clustered as Web Traffic. However, if a non-known port is used for the connection then the flow is clustered in the generic TCP, UDP or ICMP traffic. After the k-mean clustering algorithm is applied, clusters of flows are created based on their assigned service. In order to define if a cluster is normal or abnormal the authors decide to use heuristics such as the average number in packets instead of simply the size of a cluster. After that, the classification of new incoming flows as normal or abnormal is based on two different factors. The first one is the distance of the new traffic from all the cluster centroids and the classification is based on the label of the closest cluster. The second one is the outlier detection where new traffic is labeled based on the distance from normal labeled clusters only. Finally the authors suggest that a combined approach will yield the best results. On the evaluation aspect, the authors create two datasets a testbed with a known flood attack injected and a dataset from a student residential network from the University of Twente. As expected the first dataset exhibits a good performance and is able to identify the flood attack based on the network features monitored. However, the second dataset that is closer to real network traffic has a few challenges. Because of the simplistic approach on the service labeling and only considering the known ports, most of the traffic was unknown since high number ports are used nowadays by many applications. This resulted in traffic generated by applications to be classified as abnormal because of the high amount of traffic generated. In general, the authors present the feasibility of K-means clustering which is considered an expensive algorithm resource-wise and the obstacles they face give motivation for the novelty of the approach on the thesis.

3.3 User Network Behavior Analysis

The research methods in which application traffic can be classified inside a vast amount of network traffic have been presented. Moreover, the concept of anomaly detection on a network level has been analyzed and the possible approaches have been presented. The final concept of the thesis is the user network behavior aspect and how from a general anomaly detection perspective the focus can be directed into the actual user behavior. There has not been a great deal of research work on user network behavior combined with anomaly detection but this only gives more motivation to pursue this aspect.

Before presenting the user network behavior aspect, the work of G. Wu et al [45] gives a better understanding of the application network behavior and its consistency. The authors propose a framework to capture and characterize the network behavior of applications in order to validate their behavioral consistency. The authors suggest that normal applications that fall into the same general category e.g. social media, music, news will exhibit similar network behavior. Their goal is to identify malware applications by observing deviations when compared to the normal behavior model of the category in which they claim to be. The suggested architecture consists of a testbed platform that will generate the application network traffic and a machine learning model that will receive the traffic as training input. The testbed component is a scenario generator that simulates all application related actions which are UI interactions, application changes and hardware changes. By monitoring and capturing the network traffic generated when applying a random combination of these scenarios the training dataset is created. To perform the experiment, the author used a group of the mostly downloaded android application on the Chinese download platform HiAPK as well as a number of known malware applications. The network features that are extracting from the traffic are the number of bytes sent and received, the number of concurrent connections during each test scenario as well as the connection duration. Afterwards, the extracted network features are fed into a recurrent neural network (RNN) that tries to model the application behavior. The results of the research are quite promising since malware applications exhibit a behavior that is greatly different from the normal app behavior. Moreover, the behavior models consistency is acceptable although the scale of the experiments and the variety of applications is not that broad. Such results provide insight on the expected application behaviors on the network and their consistency and is a good guideline for our own research.

From a user network behavior aspect, the core idea of such a model has been presented on a high level by Kim et al [46]. The authors describe a framework that collects the network traffic of users inside a BYOD environment and forwards it to a user analysis system. The network traffic gathered from each device is used to reflect the user behavior and contains contextual information such as device used, time of connection and type of application traffic generated. All these contextual information are used to craft a user profile based on which a normal behavior is created. When the user diverges from the normal behavior model created specifically for him then an abnormal behavior alert is generated. Although, the suggested model is on a very high level and no specific details are presented, this core idea is closely related to the thesis.

Zhang et al [47] approach anomaly detection on user network behavior through cross

entropy analysis. The authors focus on user behavior on a group level and not on a per user scale and thus consider a user flow which is the aggregated behavior of all the users in the network. The proposed method is to analyze the distribution of user flow over time in order to classify monitored days into normal and abnormal ones which are weekend and holiday. The user behavior distribution is calculated through the entropy of the network traffic which is expected to be higher in abnormal days since the human behavior on the network is not as static as a normal day of the week. To evaluate the method, a dataset of WiFi network traffic was collected in three different scenarios, a university campus, an office and a hotel. The dataset consists of the MAC address of a device and a timestamp of the connection time. The authors propose two models based on cross entropy to calculate the distribution of traffic, a simple cross entropy method to split the traffic and an enhanced one that uses fuzzy logic to classify traffic more accurately. After training the model and trying various values for the classification threshold the authors were able to successfully identify anomalies based on the high entropy off-days presented. Both models exhibit a distinct shift in the user network behavior over the weekend and holidays compared to normal weekdays.

An anomaly detection model of network user behavior based on the Artificial Immunity System (AIS) [48] is suggested by Zhang et al [49]. Because of the AIS characteristics of self-adaptation, immune feedback and diversity the authors believe it will be suitable to detect the dynamic behavior of users. In order for the model to work, all the network elements need to be translated into the parts of an AIS system. For this reason, normal and abnormal behavior, a detector and a simulated network dataset need to be provided to the AIS system. For the purpose of the research, the authors simulated a network of a few computers visiting random websites for a few hours and then configured five of the machines to perform a varying size flood attack. From each packet of the network traffic that was generated, a signature from the packet header was extracted and transformed into a binary string. These strings, both normal and abnormal, were fed into the AIS and a detector was trained. Although the model has a small scale dataset, the false alarm rate is close to 5% and the model introduces a new perspective for user network behavior analysis.

Wei et al [50] propose a user network behavior model through a scenario based graph analysis approach. The authors create a graph representation of the user browser actions in order to model his behavior. Despite the large number of identifiers on a browser connection between a user and a server, most of them exist only during the initialization of the connection. However, a session id cookie is persistent and most of the times used to keep a connection alive. Therefore, the authors suggest that a session id is the suitable string to be Fused to bind a network flow to an application. The graph model consists of nodes which are the resources used during the user browsing. Such resources are the response data of the server and include images, ad traffic and raw application data. Resource nodes are connected by edges that represent the user action e.g click, login and have a corresponding session id attached to them. Consequently, a user behavior pattern is the sequence of nodes that are connected through edges with the same session id. After creating the user behavior patterns, the authors simulated a few scenario around browser actions. After the creation of the graph based on those scenarios, a clustering algorithm was applied to create groups of resource nodes that are close enough and are considered part of the same session connection. To characterize a user behavior pattern as a specific scenario, the user behavior

patterns are compared to the clusters of resource nodes by comparing the contained nodes in each element. For the evaluation of the approach, a simulation website was made with four basic functions agenda management, personnel management, system management and document management. The suggested approach is able to identify simple scenarios with a high success rate but has less success against scenarios that create noisy traffic. A graph based approach on user behavior can provide insight on specific user patterns but lacks the ability to properly scale for multiple applications as well as the ability to cope with noisy traffic.

Kang et al [51] propose a anomaly detection model based on user behavior by using Bayesian Inference. In order to create the model, the authors decide to focus on information that are not directly related to network information. Their dataset consists of user behavior information such as duration of connection, time of connection, location of the user, type of device used which are considered as a sequence of actions. The behavior model that uses this type of information is based on Bayesian Theory and tries to calculate the probability of an action occurring by taking into consideration the actions before that one. In such a way, a probability matrix is calculated for each action and used to detect abnormal behavior. Each row in the probability matrix denotes the chances of an action occurring if another one has occurred before. Therefore, by observing the sequence of the aforementioned user behavior actions the values in the matrix can be updated to try and predict the behavior. If a sequence of actions has a low probability value then it is classified as abnormal. As the authors state a framework that models human behavior is challenging since humans can be unpredictable. In their evaluation phase, the authors created a virtual enterprise business system with a total of 5 users with 500 connections each to train the model. However, the final results display a large degree of variance on the behavior and therefore the probability matrix is not reliable enough.

Vaarandi et al [52] propose two algorithms to detect anomalous traffic in an organizational private network by creating user behavior profiles. Both of their suggestions are based on the services used by an employee inside the network. As service the authors define the tuple of information (IP address, port number, transport protocol ID). Moreover, the authors performed a service analysis on the network to get a better understanding of what services exist, the frequency of their usage and the amount of active users. The dataset consists of NetFlow data from a private backbone network of a large financial institution. Both of the suggested algorithms utilize a service detection method to identify TCP and UDP based network services from the recent past. This information is aggregated and used to generate user behavior profiles. The first algorithm is based on the frequency of a service in a user profile and how likely it is to reappear in the future. Therefore, when a new service appears that has a low frequency or has never emerged in the user profile before it is considered abnormal. An interesting finding by the authors is that several services while being widely used in the organization displayed a varying frequency on each individual profile. Based on them, such services should also be included the user profile and not be mistaken as abnormal. As a result, the combination of a user profile and a list of widely used services is compared with new services in order to classify them. The second algorithm proposed by the authors is a clustering algorithm that tries to combine user profiles that exhibit similar service usage. When a cluster member demonstrates a new service usage and it does not correlate with

the rest of the members service activity then it is considered abnormal. With these two approach the authors were able to identify misconfigured devices, rogue SSH connections, anomalous end user actions as well as a number of cases where cluster members did not consume a service used by the rest of the nodes in the cluster. This research is closely related to our approach with a difference on the scope of network traffic. In this thesis, the focus is on services on an application level and the combination of network flows if they are part of the same application traffic. In such a way, better insight on the user behavioral patterns as well as a more accurate anomaly detection can be potentially achieved.

3.4 Related Work Overview

From the mobile application identification perspective, both presented approaches have certain limitations. However, the attribute of scaling is an important one when it comes to an organization network. Therefore the payload-based approach is selected for this research and an identification algorithm that correlates network traffic to specific mobile applications is introduced. Moreover, the algorithm tackles an important obstacle mentioned in related work, the omission of encrypted traffic with a novel string similarity based approach.
The aspect of user behavior on a network level can be challenging since the creation of such a model is not an easy task. As mentioned by various researchers, their proposed machine learning algorithms have difficulties to create a model to reflect user behavior because of its dynamic nature. This translates into data that have a high variance and are not easily grouped together into models that are not sufficiently robust. More specifically on the sector of clustering algorithms which is the focus of the thesis, related work has applied almost exclusively algorithms such as k-means that are prone to errors when it comes to noise and outliers. Moreover, such algorithms pre-define the number of clusters before execution which limits the output efficiency. Finally, most of the focus has been on supervised clustering algorithms that require a labeled dataset and focuses on known network attacks instead not the human related ones. Thus, there is room for improvement and lack of applied work in the sector of anomaly detection and more importantly on the utilization of unsupervised machine learning techniques. All of these points are the inspiration and guideline for the decisions made in the proposed framework. The work conducted on this thesis tries to fill that gap and provide a different more flexible approach on clustering with the goal of behavior analysis and anomaly detection. Moreover, the proposed framework is evaluated using real scenarios of user behavior related anomalies so as to prove the feasibility of such an approach.

Chapter 4

Dataset Collection & Analysis

The goal of the proposed framework is to analyze network traffic related to mobile applications and utilize it to train a clustering algorithm that will reflect user network behavior. However, raw network traffic has an abundance of information a part of which is unrelated with the goal of the thesis and therefore out of scope. When it comes to machine learning algorithms, a well defined and formulated dataset is an important part of the algorithm efficiency. For this reason, the dataset has to go through different filtering processes. Moreover, the application identification algorithm is applied to separate the traffic into meaningful sets of information based on the application to which they belong. Finally, an analysis of the state of the network traffic after these steps and possible challenges as stated in related work are explored. An overview of the steps described in this chapter are presented in Figure 4.1.

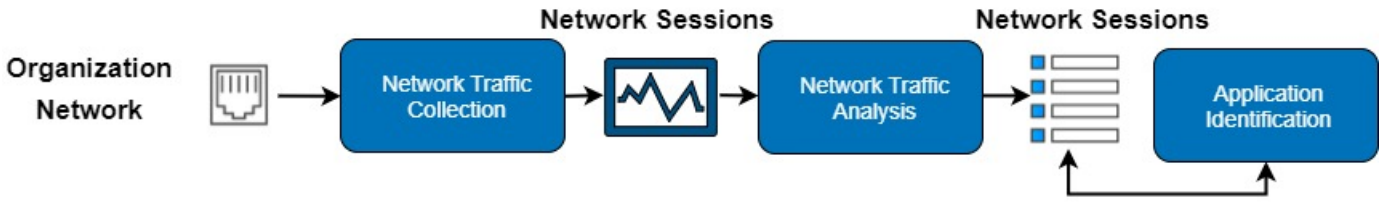

Figure 4.1: Network Traffic Collection and Filtering

4.1 Network Traffic Collection

For the purpose of the research, the network traffic from a large private organization was captured for a period of 2 weeks. The network traffic was captured and stored with the use of port mirroring and included exclusively the Wi-Fi traffic generated by company employees mobile devices. This resulted on a daily average of 1480 unique internal IP addresses and a total of 3.2TB monitored network traffic throughout the 2 weeks. All the internal IP addresses were properly anonymized by hashing each address combined with salt before usage so that there can be no connection between an IP address and a specific employee.

However, the network traffic generated by users is initially in raw pcap format. An analysis on a network packet level would provide low level network information, e.g, traffic bandwidth based on assigned intervals which is not closely related to the concept of user actions on the network. Such an approach would be preferred if we were to monitor the network and identify generic changes in traffic which is not the case in the thesis. This means that it is preferable for packets to be aggregated on meaningful groups and their combined network features used to extract information from the network. Therefore, a suitable format of the dataset that can facilitate the analysis on a level higher than the network packet one is required. For this purpose, the concept of network sessions is introduced. A session object is defined as a stream of IP packets aggregated based on a shared IP five-tuple (protocol, source IP address, source IP port, destination IP address, destination IP port). Based on the network characteristics and headers of the aggregated packets, each session object is an instance of a connection between a user and a mobile application and contains general network information. The sessions provide a good reflection of user actions on the network and their network features can prove to be useful as features for the proposed model. Such information are the number of packets received and sent, the duration of the session as well as the transport layer protocol.

Moreover, each application level protocol has its own established rules and requires the exchange of clearly defined information between the parties during a connection. All of this information construct a rich set of data regarding the network traffic and will be extracted from the pcap file and embedded in each session so as to be used in the application identification algorithm. More specifically in the case of encrypted traffic, application related information exchanged by the two parties are required such as SSL certificates or specific header fields. When it comes to unecrypted traffic, HTTP related information available through the packet headers are of interest. Finally, the DNS protocol provides useful information when it comes to correlating IP addresses to known applications.

Therefore a tool that will transform the pcap file from a set of packets into sessions and provide all the aforementioned information is required. After examining the available network monitor frameworks that provide this functionality, the open source network analysis framework Bro [53] is chosen. Bro provides detailed analyzers regarding many protocols such as HTTP, SSL, DNS. When the framework is provided with a pcap file as input, the file is parsed and network sessions are detected and created. For each network session based on its network protocol detailed information are stored in multiple logging files different for each protocol.

4.2 Network Traffic Filtering

When it comes to network traffic in an large organization, various protocols and type of sessions are expected to be found in the network. This translates into network traffic that includes many different network protocols either on an application or transport layer. Since the scope of the research is on a mobile application level, a fitting filtering of the sessions is required to remove unrelated traffic. By keeping only traffic that is expected to originate from a mobile application we facilitate the application identification algorithm and

minimize the cases where a flow is misclassified or remains unknown.

First of all, on a transport layer level the only protocols that include traffic which is related to mobile application are the TCP, UDP and QUIC protocol. Although the vast majority of applications utilize the TCP protocol for their connections, some applications use the UDP and QUIC protocol and therefore they are included. A number of ICMP packets were discovered in the monitored traffic but based on the ICMP protocol usage they are not expected to contribute information meaningful to a behavior analysis concept. As a result, only the TCP, UDP and QUIC network sessions are taken into consideration for our dataset.

Moving to the application layer, there are various protocols each one defined for a different purpose. The Internet Assigned Numbers Authority (IANA) is responsible for maintaining the official assignments of port numbers for specific uses. Network ports are separated in three categories regarding their use based on their values. The three categories are:

- **Well known ports**: The port numbers range from 0 to 1023. They are used by system processes that provide widely used types of network services.
- **Registered ports:** The port numbers range from 1024 to 49151. They are assigned by IANA for specific service upon application by a requesting entity.
- **Dynamic, private ports:** The port numbers range from 49152 to 65535. This range is used for private, or customized services or temporary purposes.

After examining the purpose and usage of services in the well known port range, it is clear that a large amount of them are not related to mobile applications. As a result such traffic can be excluded to further clear the final dataset. After an analysis on the destination ports and different services found in our dataset, the vast majority of traffic related to mobile applications has as destination the ports 80 and 443. Apart from these two, the only other ports used that were related to encrypted versions of mailing services. The protocols and the port numbers in that range that are useful for the research are presented in table 4.1. In addition, more and more mobile applications utilize ports on the two other range values for their custom tailored functionalities. A few examples of known applications found in the analyzed traffic are WhatsApp which uses the protocol XMPP and port 5222 for its connections, Facebook that uses sometimes port 3478 for STUN protocol related communication services and Spotify that occupies port 4070 for its connections. In order to avoid excluding network traffic that can provide information regarding mobile applications all ports on the registered and dynamic port range values are accepted. As a result, sessions that do not have a destination port included in table 4.1, in the range of registered port or in the range private ports are filtered out.

Finally, the duration of a session is considered as whether the session should be included in the dataset or not. The phenomena of very short connections with a small amount of traffic can negatively affect the dataset during the analysis if the information they contain are not actually related to user actions. Scenarios of such short connections are when a device is trying to establish a connection with a server but the connection is dropped prematurely or reset which means that this traffic is not important for the analysis and can be discarded. However, there are a few cases of short connections that could provide useful

Table 4.1: Application Layer Protocols

Protocol Name	**Protocol Port**
smtp	25, 465
http	80
ssl	443
quic	443
imap (over ssl)	993
pop3 (over ssl)	995

information such as app API calls. The Bro framework provides alongside each session a state field based on the content of specific communication protocol packets. The state gives an overview of why the connection ended, e.g, server never responded to client request or server responded but client dropped the connection. To identify which of these scenarios actually occur and whether it is safe to exclude such malformed session, an analysis is performed on the captured traffic.

For every day of captured traffic, an average of 1,253,512 network sessions are recorded. In order to decide on a session duration threshold, an analysis on the dataset was performed based on the duration and the Bro state field. The threshold that gave the minimum number of normal connections discarded and the maximum number of malformed sessions dropped is 500 milliseconds. To further verify what type of traffic is exactly excluded based on this threshold, all the sessions with a duration less than 500 milliseconds are extracted from the dataset. This resulted in a total of of 984,239 sessions which sums up to a 7.8% of our total collected traffic. Out of these sessions, 836,603 (85.0%) are malformed sessions based on Bro provided states and 147,636 (15.0%) are session with normal establishment and termination.

Therefore the majority of short sessions are indeed connections that were not properly established. However, to verify that no useful information is discarded, we look for repeating patterns in the destination IP addresses of these 147,636 sessions that would indicate the use of an application specific functionality on such a short time. The sessions had a total of 35,348 unique destination IP addresses. To understand how often the same IP address appeared on a session the distribution of sessions belonging to a specific destination IP address are presented in table 4.2. The overwhelming majority of sessions refer to a destination IP address only once. Moreover, there are a few addresses that reappear on a rather small amount of sessions but after manual inspection most of them refer to revisiting websites. Finally, two specific addresses stand out because they appear very often on sessions. After manual analysis, the two destination IP addresses that highly stand out belong to specific company-related services on the employee devices which are not stated for privacy reasons. Considering the distribution, we can see that the dropped sessions that are not malformed do not hold an important number of information that could harm the results of the thesis. Overall, it is demonstrated that the duration threshold of 500 milliseconds is suitable to remove unwanted sessions without losing information related to mobile applications that could prove to be useful in the next steps.

Table 4.2: Session/ Destination IP address Distribution

# of sessions per IP	1	2-5	6-10	100+
# of IP addresses	19,020	12681	3643	2

4.3 Application Identification Algorithm

With the network traffic transformed into sessions and initial filtering steps applied, the actual mobile application to which they belong needs to be identified. As described in the related work chapter, a payload-based approach is chosen for this research since its the most suitable for scaling on a large organization. An overview of the proposed approach is presented in Figure 4.2.

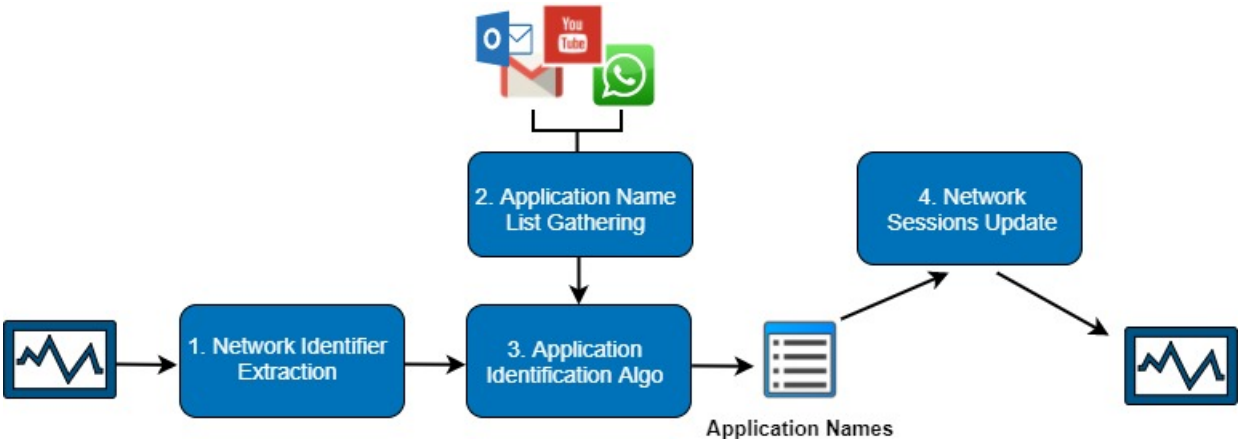

Figure 4.2: Application Identification Overview

In order to identify the application, we will use information from the network sessions. As mentioned earlier, different type of information, based on the connection protocol, exchanged between the two parties of a connection are extracted. More specifically, we decide upon specific string identifiers from this information that will match a network flow to an application. In our approach, we choose to find identifiers that will reveal the name of the mobile application that "owns" a network session. For this reason, an analysis on the fields and rules of the SSL and HTTP protocol is performed so as to determine such elements for encrypted and unencrypted traffic accordingly. Both protocols exchange information during the establishment of a connection between a client and a server which are used to exchange configurations and details about the connection. Therefore, we focus on this piece of data to extract identifiers that will reveal the application.

At the start of an SSL connection, a handshake takes place where the two parties exchange information regarding their identity, the purpose of the connection and other technical details before starting to exchange encrypted information. During that handshake, this unencrypted flow of information is examined in detail to identify the mobile application. To get a better understanding, an example of the SSL handshake of a Facebook SSL connec-

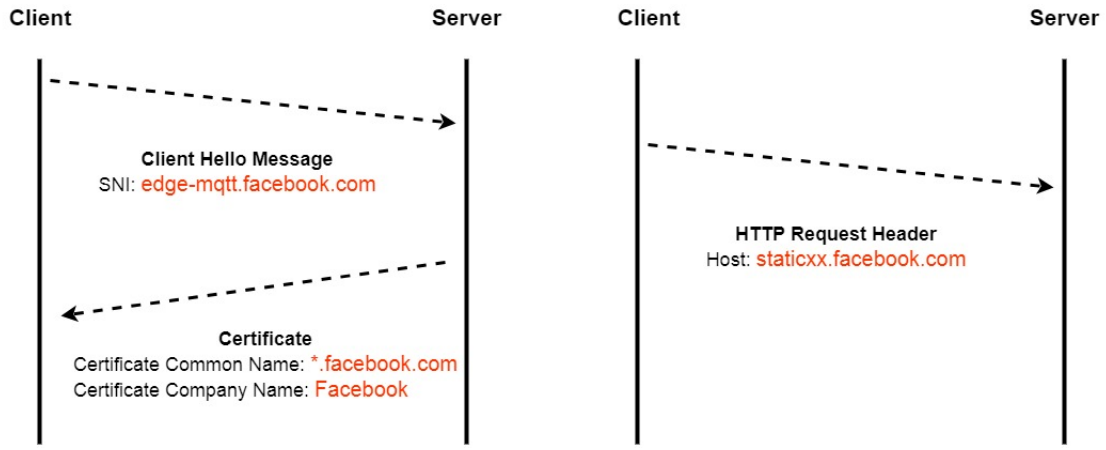

Figure 4.3: HTTP / SSL Handshake Information

tion extracted from our dataset is presented in Figure 4.3. To initiate an SSL connection, the client has to send a *Client Hello* packet to the server containing information regarding the configuration of the connection he wants to establish. Based on RFC 6066, as part of this packet the client is able to include a field called *Server Name Indication* (SNI) to indicate the hostname the client is trying to connect to. Although the field is not mandatory, only 2.8% of the monitored SSL connections did not include a value for the field. Moreover, as subsequent parts of the SSL handshake the server sends a packet including SSL certificates that are related to the cryptographic aspect of the connection but also include interesting information. Typically, the server sends a chain of certificates where the first one of the chain is the one directly related to the user-requested hostname. In each certificate there are two fields, the *Common Name* and the *Company Name*, that can help us identify the mobile underlying application. As seen in Figure 4.3, all three fields contain information related to the Facebook application.

When it comes to the HTTP protocol that does not include encryption, the process is simplified. In this case, the information exchanged between the two parties, before establishing a connection, are less. A specific header in the HTTP packet that defines the domain server is of interest in this case. The Host request-header field specifies the domain name of the server being contacted as obtained from the original URI given by the client. A similar case of an HTTP connecting to Facebook is also presented in Figure 4.3. Finally, in cases where neither of the aforementioned protocols yield useful information, the DNS protocol information are utilized. The DNS protocol is used to translate domain names to IP addresses. This means that when a user requests a specific domain name it is first translated into an IP address before other protocol specific actions take place. This name value is transmitted unencrypted through the network and can be used for the purpose of the algorithm. A problem with DNS it the fact that caching mechanisms can prevent the transmission of such information on the network during a connection. However, the cases where the information from SSL and HTTP are not available are extremely rare. To summarize the network protocol fields chosen for the application identification algorithm are:

- Server Name Indication (SNI)
- Certificate Common Name, Certificate Company Name
- Host request header
- DNS Query Name

Once these identifiers are extracted from every network session, the actual mobile application needs to be identified. Although this process can be easily performed manually by a person since the names of applications are easy to identify in a string an automated scaling approach is required. For this purpose, a list of popular mobile applications from varying categories are collected from different Internet sources e.g, websites, reports with the use of custom scripts. The result is a list of 3000 known mobile applications that cover a big part of the spectrum of app categories and mobile operating systems. Although, many of these applications are not expected to be found in an organization network since most employees tend to install only a handful of the most famous apps such as Facebook or Gmail the list provides a certainty that cases were applications are not identified are minimized.

The extracted identifiers as well as the list of application names are provided as input to a string similarity algorithm. A pseudocode Python interpretation of the algorithm is presented in Figure 4.4.

For the string comparison technique, there are various different algorithms that can be used to measure the similarity between two strings. However, as seen from the examples in Figure 4.3, the strings under comparison are not always only containing the name of an application but also have attached other technical related information. However, the notion behind our string comparison is whether the application name is included partially of fully in the extracted network string identifiers. Certain algorithms set some limitations that do not work well with this concept. For example, the Hamming distance metric requires both strings to be of the same size which is rarely the case. Moreover, cosine and Jaccard distance metrics compare only the content of the strings without considering the order of appearance. With such limitations in mind, the string comparison algorithm chosen for our purpose is the Levenshtein distance which counts the minimal number of insertions, deletions and replacements needed for transforming one string to the one it is being compared to.

More specifically the Python library FuzzyWuzzy [54] is used which offers an implementation of the distance metric. When provided with a query string and a list of available strings the library returns the most similar string from the list as well as a similarity score. The similarity score value ranges from 0 to 100, where a score equal to 100 indicates that a perfect match has been found. Moreover, we need to define the similarity score below which the string returned from the list is not similar to a satisfying degree. For this purpose, the similarity scores of the network identifiers of the two week network traffic are stored and analyzed. In Figure 4.5, the distribution of the similarity scores is presented. The majority of comparisons yield a score of 100 and these perfect matches refer to all the well known mobile apps. As the similarity score drops, more cases of known apps resources such as CDNs appear and still have a score close to 100. However, once the similarity score dropped below 90 the responses matched with the query string solely based on the fact that they shared a few characters without them being truly related. Most of these cases refer

```
def levenh_algo(identifier, application_name_db):
  # Single Comparison
  [candidate, score1] = Fuzzy.compare(identifier, application_name_db)

  # Tokenized Comparison and keep max score2
  for tokID in identifier.split("."):
    [candidate, score2] = Fuzzy.compare(tokID, application_name_db)

  # Tuple Tokenized Comparison and keep max score3
  for tuple in combinations(identifier.split("."), 2):
    [candidate, score3] = Fuzzy.compare(tokID, application_name_db)

  return max(score1, score2, score3)


# Application Identification Algorithm

# identifier_list = [SNI, Certificate Fields, Host Name, DNS Name]
# application_name_db: All the application names scrapped from the Web
for identifier in identifier_list:
    (candidate_name, similarity) = levenh_algo(identifier, application_name_db)

    if similarity == 100:
      return candidate_name
    else:
      candidates.append([candidate_name, similarity])

    # Returns the candidate with the highest similarity score
    # and highest frequency in case of equal scores.
    [best_candidate, best_score] = best_score_frequency(candidates)

    # If the best candidate is not suitable
    # The longest common substring is extracted.
    if best_score<=90:

        extracted_ID = LCS(candidates)
        return extracted_ID
    else:
        return best_candidate
```

Figure 4.4: Application Identification Algorithm

to website URLs whose host name happen to share characters with various applications. However, since the goal is not to identify random websites visited through a mobile device but rather mobile applications these cases do not contribute to the analysis. Therefore, a similarity score below 90 implies that there is no satisfying similarity candidate.

For our algorithm, each of the extracted identifiers is compared against the collected mobile application names. Since the strings under analysis do not contain the application name in a straightforward way but include also other pieces of unrelated information, a few modified versions of the provided identifiers are compared against the list. In this way, the cases of algorithm false classification are minimized. Initially, the string without any alterations is compared in order to capture simple cases where the session explicitly defines the target application. However, in many cases communication with application resources such as servers include addresses and domain names with randomly generated ids or extra information. For this reason, the provided identifiers are split into tokens based on the dot value "." and each token is individually compared against the list. Moreover to capture

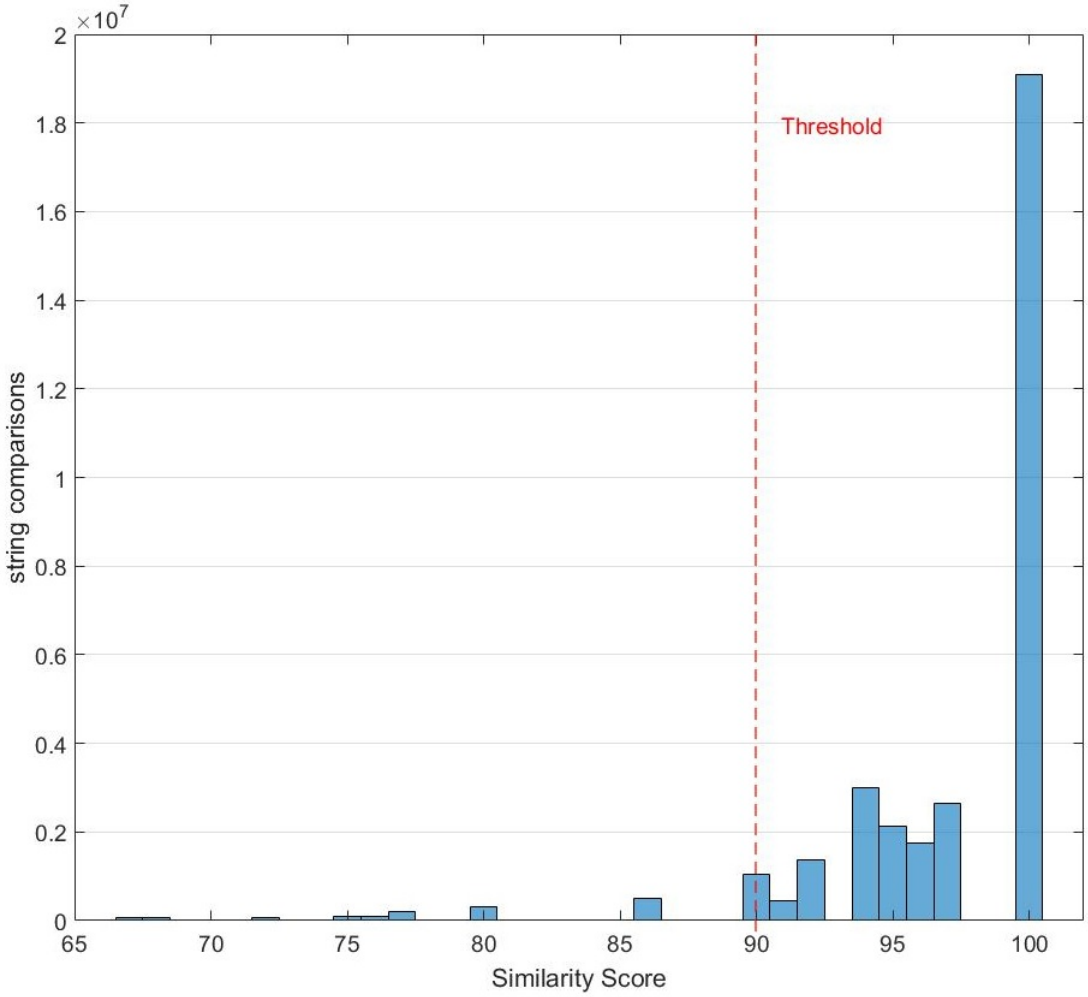

Figure 4.5: Similarity Score Distribution

cases where an application name consists of two words, tuple combinations of the tokens are also compared against the list. Although this number of comparisons increases the approach complexity, it also increases the efficiency and accuracy of detecting applications which is the main concern. If none of the comparisons yields a perfect candidate then the candidate with the maximum similarity score is kept. If the chosen candidate has a score equal or lower than 90, this means that there is no suitable application name for this network session. So as to not falsely classify the session into an application but at the same time keep an identifier for potential future manual analysis the algorithm of Longest Common Substring (LCS) is applied on the list of identifiers. The result is stored in the session object instead of an application name and with this approach a manual inspection of such cases can be performed to identify the application or the scenario where the algorithm is unable to detect one. Although, such scenarios of manual analysis on extracted identifiers were not

Table 4.3: YouTube Session Identifiers

Identifier Name	**Identifier Value**
SNI	m.youtube.com
Certificate Common Name	*.google.com
Company Name	Google Inc

Table 4.4: Instagram Session Identifiers

Identifier Name	**Identifier Value**
SNI	instagram.fgua2-1.fna.fbcdn.net
Certificate Common Name	*.fgua2-1.fna.fbcdn.net
Company Name	Facebook

used in our thesis since the majority of them refer to websites this part is included for the completeness of the algorithm.

Moreover, an important part of the algorithm is the order in which the identifiers are compared against the known application list. In the case of encrypted traffic, the protocol specific identifiers are the SNI and the certificate ones. Despite the fact that all identifiers belong to a session regarding the same application, the level of detail on their content can differ from case to case. In general the SNI and Host request identifiers are the most precise ones when it comes to including the application name. On the other hand, when companies issue certificates the related identifiers can include string elements such as an asterisk (*) to capture a larger scope of network domains or the higher level organization name that owns the application. One example is the details of a YouTube session presented in table 4.3. In this example, the SNI field explicitly declares the application under usage but the certificate is on a larger scope so as to cover general Google related services.

Apart from simply missing pieces of information, there can be cases where the more generic identifiers return a perfect match from the database and the algorithm terminates believing that the best candidate was found. However, this is a case of misclassification and an example from our traffic to explain this scenario is presented in Table 4.4. These are the identifiers of an Instagram session and in this case the Instagram application is owned by Facebook. It can be seen that the SNI identifier clearly has a token with the correct value. However, if we were to first compare the Company Name against the list of collected application names then the session would have been falsely considered part of the Facebook application traffic. To avoid such cases, the identifiers are compared against the known application names starting with the most precise identifiers and moving towards the generic ones.

4.4 Network Traffic Analysis

All the actions regarding the filtering and processing of the monitored network traffic have been described in detail. These steps have the goal to optimize the format of the dataset and provide a final filtered version that will exclude unrelated to mobile applications traffic. Before moving on to the next chapter regarding the clustering part of the thesis and the reasoning behind choises made around it, we first analyze two main arguments regarding the network traffic and its scope. First, we provide insight on how the separation of traffic on an application level is beneficial and will help the clustering algorithm. Moreover, the noisy nature of network traffic that is mentioned by researchers in related work is verified and as a result this provides further motivation and reasoning for the choice of clustering in the following chapter.

4.4.1 Session Separation Scope

As mentioned before, traffic is separated based on the mobile application that generated each network session. However, in other research approaches the session separation is either on a destination port level [44] or the IP five-tuple [52]. In approaches like the first one, sessions that belong to different applications will be clustered together solely based on the fact that they use the same network protocol. However, as mentioned by the researchers themselves this led to a excessively broad scope of aggregation and the inability to detect many specific patterns or anomalies. Furthermore, the second approach is not able to identify the relationship between sessions that have a different IP address or port. Consider mobile applications with various functionalities where each functionality initiates a connection using a different port or IP address. These cases would go unnoticed and potentially useful information will be lost. For these reasons, in the thesis the sessions are separated on a mobile application level. With the proposed method, all of these sessions will be considered in the same group since they serve the same mobile app. In this way, all the notions of user related actions reflected on the traffic will be captured and a more well defined analysis is performed.

When the goal of the framework is to identify patterns and anomalies based on the user actions, our detailed scope will facilitate the process. If sessions are aggregated on a very high level, the clustering algorithm will have a harder time to cluster sessions or will create not representative clusters. Moreover, different applications are expected to exhibit different behavior on a network feature level. This means that a group of sessions that are considered normal for one application can potentially be considerably over the value range of another one. This can cause great problems for the clustering algorithm and again result in malformed clusters that are not the best representation of the user actions and behavior.

To better understanding this problem and the proposed solution, a visualization of network traffic first on a TCP level and then on a per mobile application level is presented in Figure 4.6. A sample traffic of a random employee of the organization is used. In this case, the traffic is displayed through the session features of duration and number of kilobytes sent from the user. Based on the application detection algorithm, the employee used three different applications Facebook, YouTube and WhatsApp. On the first plot, the sessions are displayed together based on the fact that they all belong to sessions with destination port

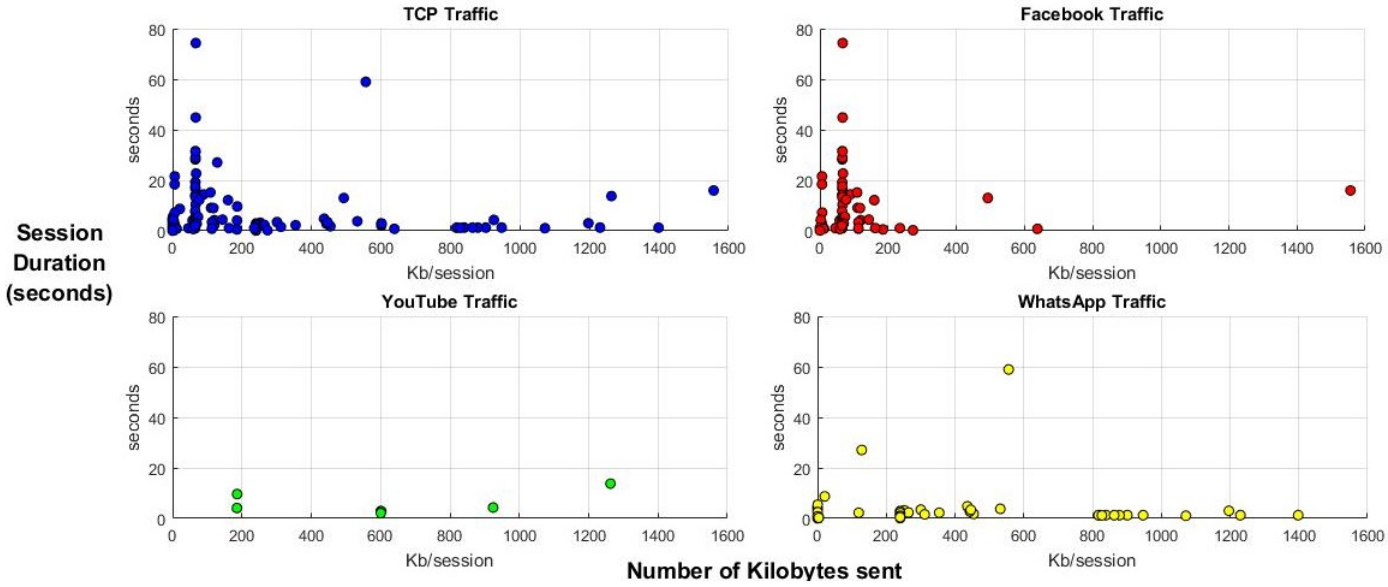

Figure 4.6: Network Sessions Separation on Application Level

443. If the clustering algorithm was to be applied at this point then several details, application specific patterns and anomalies would go unnoticed. Moreover, the spread distribution of values would create problems for a clustering algorithm when it comes to finding the optimal cluster groups.

However, in the rest three subplots, the sessions are separated based on the application to which they belong. Now a more clear overview of the application network behavior is gained as well as specific patterns when it comes to network features. In this example, each set of sessions has a more distinct behavior regarding the number of Kilobytes sent as well as the duration of a sessions. As we can see the Facebook sessions tend to have a low number of traffic and a relatively larger duration. On the other hand, traffic originated from WhatsApp has a distinctly higher value range of traffic. Finally, YouTube traffic has a more static behavior which can facilitate a clustering algorithm on the outlier detection. All of these application specific details are important in anomaly detection and would have been undervalued using the TCP level approach.

4.4.2 Traffic Noisy Nature

Based on the related work section, clustering approaches have stumbled upon obstacles relevant to how noisy and difficult to fit into a behavior model network traffic is. It is important to verify this challenge in our own dataset since it plays an important role on the selection of clustering method later on.

In data analysis, a noisy dataset is translated into a distribution of values that do not have a normal behavior and are highly spread. To identify this noisy nature in the network traffic, the standard deviation and the mean value of the main network features of sessions

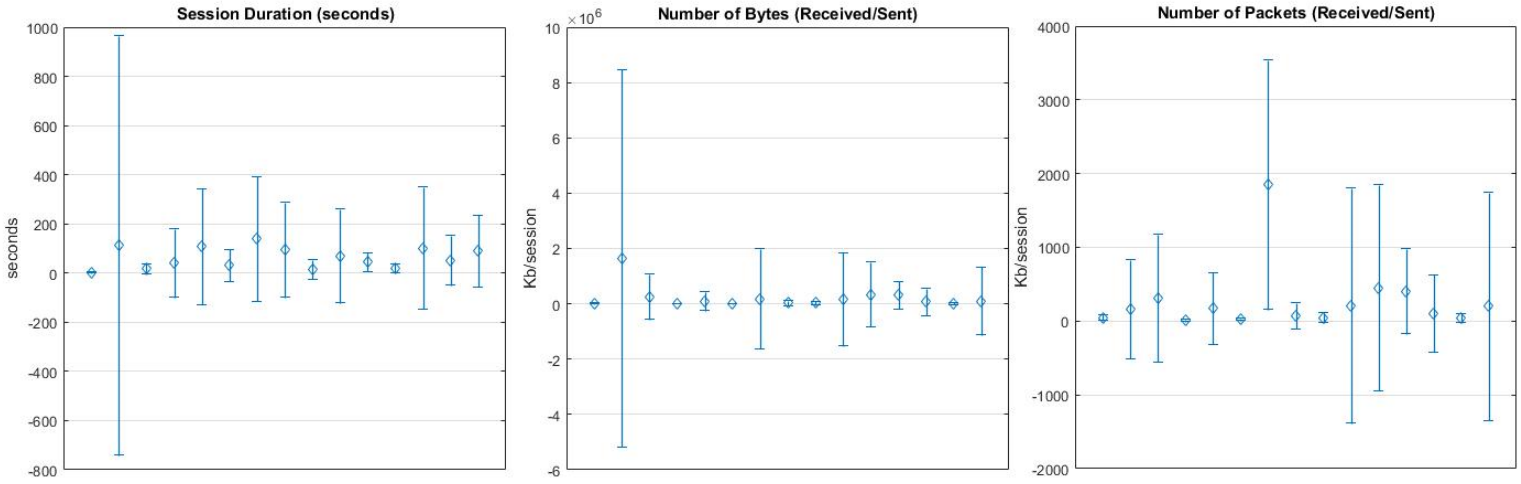

Figure 4.7: Network Feature Standard Deviation

are calculated. Moreover to provide better insight on the difference between applications traffic, the sessions are further separated based on the application they belong to. The applications chosen from the network are the 15 most popular ones based on the number of sessions each had throughout the experiment. The list includes messaging, banking, mailing and data storage applications. In Figure 4.7, the metrics for the aforementioned applications are presented through the use of an error bar that utilizes the standard deviation instead of the standard error. Each bar of the three sub plots displays the variability of the network features for each one of the fifteen applications. Therefore, a bar with high length translates into a distribution with a high standard deviation and therefore values which are not homogeneously distributed around the mean value. Most of the applications exhibit a dynamic distribution for their network features and as a result this will lead to sessions with very different values. As a result, a more rigid clustering algorithm will not be able to cluster together sessions that are spread non homogeneously on the feature space. This finding further validates the choice made regarding robust and noise flexible clustering algorithms.

From the previous analysis, it is obvious how the separation of traffic on an application level will facilitate a clustering algorithm and enhance the ability to detect patterns and anomalies. Furthermore, the noisy nature of network traffic has been established and plays an important role on the choice of clustering algorithm used. The details of the proposed framework and the method in which density based clustering algorithms are utilized is described in the following chapter.

Chapter 5

Methodology

The field of clustering in machine learning with the goal of anomaly detection and network behavior modeling offers many different algorithms. Each one has a different implementation, configuration settings and required format of input data. For the purpose of this thesis, a density based clustering algorithm is used to model network behavior and this choice will be explained in detail in this chapter. With the utilization of density clustering, the framework goal is to model user network behavior on an application level so as to detect anomalies and identify behavior changes in general. To better understand the details of the proposed framework, the usage of clustering in anomaly detection and the advantages of density based clustering are first explored.

5.1 Clustering in Anomaly Detection

As described in the related work chapter, researchers [38, 43, 44, 52] have mainly focused on two cluster characteristics and exploited them so as to classify clusters and therefore traffic as anomalous. These are the number of members of a cluster and the position of a cluster in the feature space. Since the data format in the thesis is in network sessions this is the scope under which these two characteristics will be described.

Cluster Size

The number of members in a cluster provides insight on sessions that exhibit the same behavior and highlights network pattern that appear repeatedly in the traffic. The approach in this case is that normal and anomalous traffic sessions form clusters in the feature space that greatly differ in size. The main assumption behind this approach, is that normal traffic constitutes a large portion of the total traffic. The anomalous traffic is expected to be infrequent throughout the monitoring phase and therefore when similar anomalous sessions are clustered together they create a cluster that is considerably smaller than a normal one. This can be the case when the network session separation is performed on a port or protocol level and very distinct network technical attacks want to be discovered. A visualization of such approaches is presented in Figure 5.1.

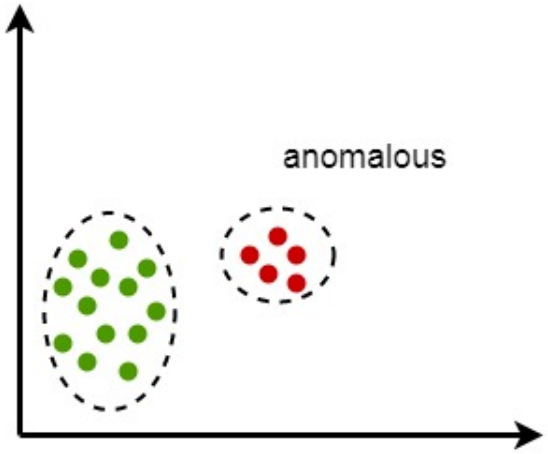

Figure 5.1: Anomalous cluster based on cluster size

However, this model has limitations regarding the scope to which sessions can be analyzed and the type of targeted anomalies. Technical attacks that follow the expected network protocols or are spread homogeneously throughout the monitoring phase can go undetected with this approach. Moreover when the main focus is not purely technical attacks but more user oriented, the assumption about what should be considered anomalous is flawed. As highlighted before, sessions that reflect user actions exhibit a high variability and are expected to be spread throughout the feature space. In such scenarios, efficient clustering algorithms will possibly create multiple small clusters instead of a few large ones. Therefore, small clusters should not be considered as anomalies based on their size but rather as specific user actions. For this reason, a model utilizing only the cluster size will have a high number of false positive when trying to detect more user oriented anomalies. Although the proposed usage of cluster size in related work has flaws it still provides vital information regarding the volume of sessions that exhibit similar behavior and will have a different utilization in our approach.

Cluster Position

A different methodology on clustering and anomaly detection focuses on the position of a cluster in the feature space. Two main approaches in which this characteristic is utilized by researchers so as to classify clusters are presented. The first one is the relative position of a cluster to the rest of the set in the feature space. When a cluster is considerably isolated from the rest it can be flagged as anomalous. In this case, the anomaly is based on the fact that the network features that caused a cluster to be isolated will have values that highly differ from the rest of the set. The second method is that a cluster is classified based on the label of the closest, in feature space, cluster. When a cluster is relatively close to an already anomalous cluster then this means that they exhibit a similar behavior.

A visualization of these two concepts is presented in Figure 5.2. In the first example, the cluster that is isolated is considered anomalous based on the fact that he is at a distance

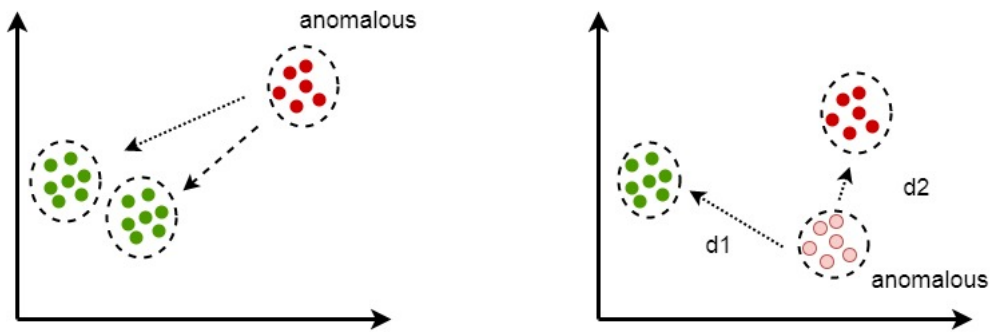

Figure 5.2: Anomalous cluster based on cluster position

from the rest. In the second example, d1 and d2 are the distance values from the clusters to be classified from the closest normal and abnormal cluster accordingly. Since the distance d2 is smaller than d1 the new cluster is also considered anomalous.

One main challenge of this concept is the performance of a clustering algorithm on highly noisy traffic and the main way for the approach to succeed is a detailed clustering that will efficiently separate traffic. If sessions are grouped in large non optimal clusters then the relative position of the cluster can potentially hide smaller anomalies caused by sub clusters inside the generic one. This type of problems can be tackled by using a precise robust clustering algorithm that will find the optimal number of clusters. Furthermore, one main disadvantage of the second methodology that measures the distance of a cluster to a pre-flagged one is the need for prior knowledge. However, clustering algorithms that require labeled data are not in the scope of the thesis. Finally as the name suggests, the notion of a cluster position in feature space needs to be defined. Moreover, the method in which clusters are compared needs to be decided upon. These configurations play an important role on the performance of the anomaly detection. In related work, various suggested methods are described such as cluster medoid distance or cluster pairwise comparison each one introducing different benefits and challenges.

5.2 Density Based Clustering

There are many families of clustering algorithms who define the notion of a cluster in a different way. Moreover, each algorithm has a distinct model that analyses the provided data in a different way and requires different type of input parameters. Each algorithm has a different output of clusters when it comes to shape and size. All these differences of the clustering algorithms make some more suitable based on the problem that has to be solved. In our thesis, we define three main requirements for each clustering algorithm to satisfy. A comparison of some of the most established and used clustering methods based on the three requirements is presented in Table 5.1. The algorithms are chosen based on the fact that they are the ones reappearing the most on related work and therefore can provide a comparison with other research approaches.

Table 5.1: Clustering Methods Comparison

Clustering Method	Defined Number of Clusters	Arbitrary Cluster Shape	Notion of Outliers
K-means	Required	Not available	Not available
Hierarchical Clustering	Required	Available	Not available
Density Based Clustering	Not Required	Available	Available

When it comes to real network traffic which exhibits a high diversity, the desirable clusters sometimes can be of arbitrary shape. This means that the points will possibly not be homogeneously spread on the feature spare and the shape of the optimal clusters that bring them together will not have always the same. Partitioning methods (k-means) and hierarchical clustering are suitable for detecting spherical-shaped and convex clusters. This means that their performance is highly affected by the solidity and clear separation of clusters in the dataset. However, as we mentioned clusters in network traffic do not meet these requirements. These are the scenarios where density based clustering algorithms outclass the rest since they are able to identify clusters of arbitrary shape.

Moreover, the noisy network traffic creates the requirement as the name suggests to characterize certain points as noise. A key aspect of density based clustering is the notion of outlier points. When a point in the dataset does not meet specific requirements regarding its distance to its neighborhood points then it is considered as an outlier. On the other hand, other type of algorithms would force all points in the dataset to eventually be assigned to a cluster even if they clearly do not belong there. When it comes to anomaly detection, this can create problems since cluster features such as cluster position and size are negatively affected by the values of outliers. This leads to the introduction of an error margin on the anomaly detection algorithm that utilizes these cluster features.

Finally, an important requirement for our clustering problem is the type of input parameters an algorithm receives. An aspect that makes density based clustering highly favorable is exactly its input parameters. Unlike other algorithms, there is no requirement to pre define the number of expected clusters, as the algorithm infers their number based on the provided data. In this way, the algorithm is not forced towards a specific solution that could not suit the data and provide false results. Similarly as before when the domain under analysis is network traffic, it is an extremely hard task to predict the number of clusters because of the high amount of traffic and its noisy nature.

Overall, based on these three requirements and the inability of many algorithms to satisfy them, density based clustering is chosen. This type of clustering method identifies areas with high density of data points and creates clusters. Points that are relatively isolated or in more sparse areas are considered outliers.

5.2.1 HDBSCAN: Density Based Clustering Algorithm

DBSCAN [55] is the most used and well defined density based clustering algorithm. However, one limitation regarding DBSCAN is the inability to detect clusters of varying density. This problem is tackled by researchers with the introduction of the HDBSCAN algorithm [56] which based on its creators is closely aligned with the concepts of Robust Single Linkage with flat cluster extraction on top of it. An analysis of HDBSCAN is presented to get a

better understanding on how the clusters are created for the rest of the thesis. The algorithm has two main configuration parameters, the minimum cluster size and the k neighbour one both of which will be explained in the process.

Initially, the concept of noise and outliers needs to be embedded in the dataset because the linkage clustering part of the algorithm can be sensitive to noise. This means that relatively isolated points could act as a bridge between two separate clusters even though the clusters should not be merged. Therefore, dense areas of data need to be conceptually brought together and outliers need to be furthered isolated. To achieve this, a metric that estimates the distance between points is introduced and called mutual reachability distance. The metric is estimated for all points in tuples and is the maximum of three compared numeric values. The first two values are the distance of each point from its k neighbour. These are called core distances and denote if the point is in a dense area or not. The third numeric value used is the distance in space between the two points under comparison. As we can understand, points with low mutual reachability distance will be in a dense area as a higher value means a point is relatively far from the rest.

The next step is to define these dense areas without having to compare all distances for each point against each other since this is extremely heavy computation-wise. For this reason, the authors utilize the concept of minimum spanning tree where the data points are vertices and an edge between any two points has a weight equal to the mutual reachability distance of those points. Moving forward, a threshold starting with a high value and continuously decreased is compared against the mutual reachability distances and used to drop edges from the graph and identify which points remain. In this way, clusters are created on each threshold value and when an edge is dropped the clusters are split in smaller ones. Therefore, a hierarchy starting with all points connected in a single cluster, since they all have a lower distance than the high threshold, up to completely isolated points is created. This hierarchy is presented with the use of a dendogram which on the leafs has isolated points and the clades are the thresholds where edges are dropped to create smaller clusters.

This is the point where HDBSCAN has a different approach than DBSCAN. In the previous version of the algorithm a, difficult to choose, parameter was required so as to cut the dendogram horizontally and select the clusters it cuts through. However, this leads to the inability of identifying clusters of varying density since a single cutoff value is chosen as the threshold. In the case of HDBSCAN, a minimum cluster size parameter is introduced and used with the dendogram so as to express the notion of a parent cluster losing points. The algorithm walks down the dendogram and on every clade checks if the split new clusters are above the minimum size. If this condition is satisfied then the algorithm continues, if not then they are considered as dropped points and the cluster retains the identify of its parent. After doing this, a condense cluster tree is created with a smaller number of nodes where each node has information about the number of points dropped through the varying distance threshold values.

The final step of the algorithm is to choose the optimal number of clusters. Based on the previous description, the dendogram provides the information of when larger clusters either split down or start to lose points because of the distance threshold. Here the authors introduce the concept of cluster stability based on which the optimal clusters are chosen. A cluster stability is interpreted as the summed time each cluster point remained in the cluster

before dropping off. As a result, clusters with a higher stability values are the ones with a lot of data points in a dense area that were truly close to each other even when a tight distance threshold is applied. Overall, as we can see HDBSCAN is a robust and flexible algorithm that tries to identify the optimal number of clusters and at the same time introduce the concept of outliers and noisy.

5.3 Proposed Framework

Now that the main characteristics of clustering in anomaly detection and an analysis of density based clustering has been presented, the proposed framework is explained in detail. An overview of the framework is presented in Figure 5.3.

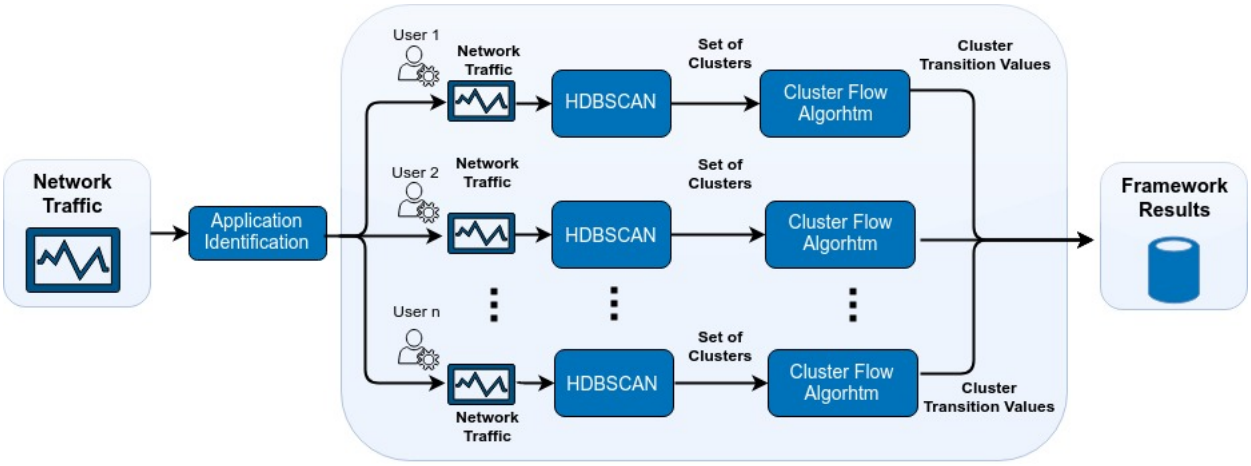

Figure 5.3: Framework Overview

5.3.1 Framework Overview

The goal of the framework is to provide a cluster flow temporal analysis of the user network behavior. The user network behavior analysis is on a per user and mobile application level for each of the monitored days. The framework output is an numeric interpretation of user behavior that will provide insight on behavioral changes and extreme shifts. To achieve this, network sessions for each user are separated into different time periods in a day and the clustering algorithm is applied. Each of the session cluster sets reflects the behavior of a user during that specific time frame. Furthermore, a comparison algorithm between these sets of clusters is applied which yields a metric, the cluster flow. This metric reflects the cluster stability and movement in the feature space throughout time. It helps us grasp the magnitude of the user behavior change between the time frames. The details of this metric as well as the cluster flow algorithm are described later in this section.

The first step of the framework is the session separation based on the application they belong to. The identification algorithm has already been described in detail in Chapter 4.

Secondly, traffic is separated on a per user level by using the internal source IP address of each session to identify which user generated the traffic. The reasoning behind this is that each employee behavior is expected to be diverse and exhibit different patterns. A separation between users will facilitate the clustering process since clusters will be able to better reflect the user specific actions and will allow more accurate predictions on their behavior. Finally as mentioned before, sessions are further separated into specific time frames to which clustering is applied. In order to identify the time frame in which a session belongs to, the timestamp of when the session initiated is used. The groups are based on the average work hours of an organization employee and the expected time frames in-between which his behavior could change. The four time frames are:

- Morning (9-11)
- Noon (11-13)
- Afternoon (13-15)
- Late Afternoon (15-17)

5.3.2 Framework Dataset

The details of the dataset consisting of network sessions that is provided as input to the clustering algorithm need to be clearly defined. The network packets that form the sessions provide the network information that will be used as features for the clustering. Moreover existing work [57, 58] on feature selection in the domain of network behavior modelling provides guidance on which features provide the most meaningful information. In Table 5.2, the selected features of each session that will be provided as input to the clustering algorithm are presented.

However, one common problem with such network features from a data analysis perspective is the highly different scale. This means that features with a higher value range will dominate the clustering algorithm and as a result have a negative impact on the outcome. Considering the high variance in the network dataset, a robust normalization technique is required. As it has been suggested by other researchers, normalization is performed by subtracting the mean and scaling to unit variance [43] or by choosing an empirical normalization factor to divide each feature [44]. However, outliers can often influence the mean and variance in a negative way and choosing a normalization factor can be a hard task. In such cases, the median and the interquartile range can be used instead of the mean and variance accordingly. The interquartile range is the difference between the upper (75%) and lower(25%) percentiles of a variable. These metrics will provide a better range of values that includes outlier points and is centered around median which is less affected from outliers than mean. For this reason, the equation used to normalize the dataset is:

$$\text{new\_feature[i]} = \frac{\text{feature[i]} - \text{median}}{\text{interquatile\_range}}$$

Table 5.2: Network Session Features

Feature Name	Feature Description
Timestamp	The time of the session start stored as epoch value
Destination Port	The destination port number of the session
Protocol Type	The protocol used in the session in protocol number
Duration	The duration of the session in seconds
Sent Bytes	The number of bytes sent by the user
Received Bytes	The number of bytes received by the user
Sent Packets	The number of packets sent by the user
Received Packets	The number of packets received by the user

5.3.3 Cluster Flow Algorithm

Once the data has been properly formulated and separated into the groups, each set of sessions is provided to HDBSCAN. The algorithm clusters the sessions together and returns a set of clusters. Each of the produced cluster sets is the network reflection of user application actions during the specific time frame. Each action e.g clicking, sending a message, sending a file generates a set of sessions of varying feature values. By using density clustering, the output is expected to be a set of clusters distinct in size and shape.

Since the goal of the framework is to efficiently monitor changes in user behavior and identify anomalies, such changes are expected to be correlated with changes and shifts between the sets of clusters on the different time frames. Possible change in the position or number of clusters in the feature space is considered as a shift in the user behavior where a drastic behavioral change should have a larger impact on the cluster sets. As a result, we need a method to quantify these changes on clusters in the feature space and identify how the cluster shift during the day. To achieve this, we introduce a novel cluster comparison algorithm that tries to identify the aforementioned elements and express them in a numeric value. In our thesis, this value is the cluster flow transition between two cluster sets. A representation of the algorithm is presented below. Moreover, a visualization of the proposed algorithm is presented in Figure 5.4.

Our algorithm receives as input two set of clusters to be compared and returns a value that expresses how different the sets are in terms of cluster size and cluster position. The value is trying to determine how the clusters have moved in the feature space and potential changes in the size. To achieve this, the process described below is applied for each cluster in both sets.

First for each cluster in the set, the closest cluster in the second set is determined. To find the closest cluster, we use the notion of medoid point to compare cluster positions. A medoid point is a cluster member that has the minimum distance from the rest of the cluster members and is therefore the most suitable representative of a cluster. In contrast with centroid, which is computed as the average value of all the cluster members, the medoid is guaranteed to be part of the cluster and inside its shape. On the other hand, in clusters with outliers a centroid is very likely to fall outside of the cluster shape and give a misleading representation of a cluster.After computing the medoid points of the clusters and

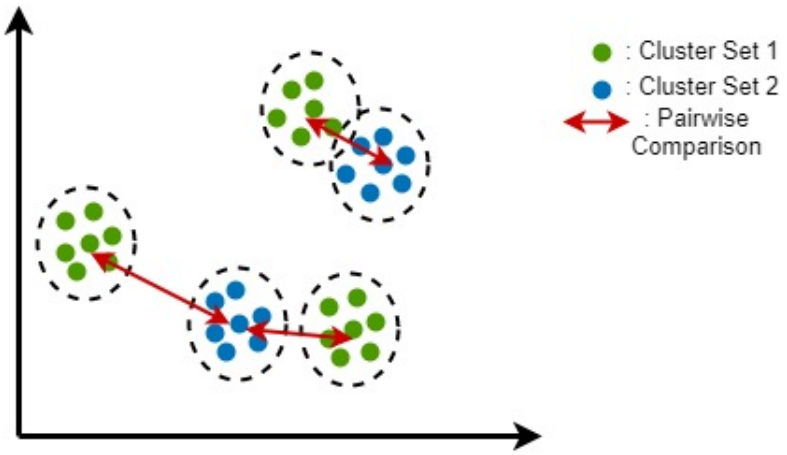

Figure 5.4: Cluster Flow Transition Evaluation

Algorithm 1 Cluster Flow Transition

```
1:  procedure ClusterComparison(set_1, set_2)
        The procedure takes the two cluster sets as arguments
        set_1 is the cluster set of the first time frame
        set_2 is the cluster set of the second time frame
2:      sum = 0
4:      for cluster1 in set1 do
5:          cluster2 = medoid_comparison(cluster1, set2)
6:          sum+ = pairwse_comparison(cluster1, cluster2)
7:      end for
8:      return sum
9:  end procedure
10:
11: cluster_flow_transition+ = ClusterComparison(set_1, set_2)
12: cluster_flow_transition+ = ClusterComparison(set_2, set_1)
```

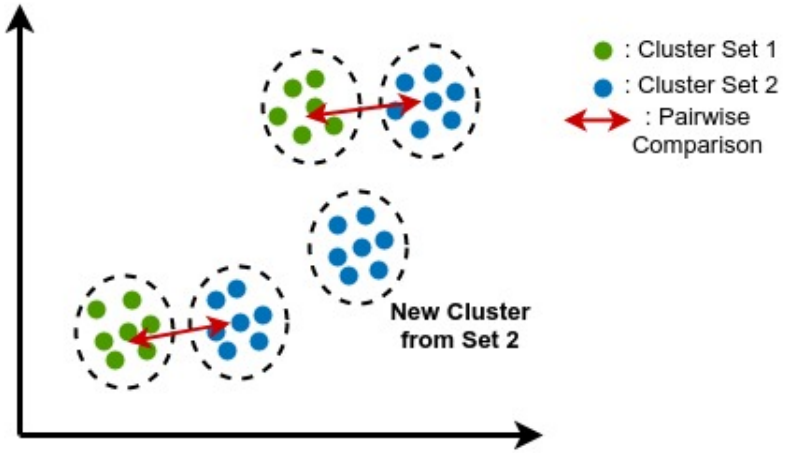

Figure 5.5: Cluster Flow Transition Example

determining the closest cluster in the second set, a pairwise comparison between them is applied. The pairwise comparison is applied by computing the euclidean distance between all cluster members of both clusters against each other so as to determine their difference in size as well as member values. This process is repeated for all the clusters in both of the sets.

An important aspect of our proposed cluster comparison algorithm is the fact that is is applied for both sets that represent the two time frames. The reasoning behind this bi-directional application of the algorithm is to capture cases where newly formed clusters are introduced. To better understand this concept, a scenario is described with the help of Figure 5.5. In the example, we consider that all the pairwise comparisons between all clusters yield a result of 1 for the sake of simplicity. Let us assume that the algorithm is only applied from the first set towards the second one as shown in Figure 5.5. In this case, the final cluster transition flow is 2. However, the fact that a new cluster is introduced in the second set goes completely unnoticed. By applying the algorithm also from the second set, the final cluster transition flow is going to be 3 since the newly created cluster is also compared against the previous set. Therefore, the information of the new cluster is better reflected in the cluster flow transition and not discarded.

The summed values of all these pairwise comparisons are considered as the cluster flow transition between the sets. Finally, the range of the cluster transition values characterizes the magnitude of the user behavior changes. With the goal of the framework in mind, the transition values are expected to reflect the user behavior on the network regarding specific applications. This means that as the transition value gets higher, the two cluster sets are considered to have an even more different representation on the feature space and therefore the user to have a more drastic change in his behavior. On the other hand, when a user has a stable behavior throughout the day the clusters are expected to exhibit the same stability on the feature space and therefore yield a relatively low transition value. Overall, the framework provides a set of values for the user daily activity on every mobile application

he used. These values help us get insights on how the application was used and whether there were any drastic changes on the user behavior.

5.4 Framework Evaluation

The proposed framework is able to quantify the user behavior changes between time frames throughout a day. In order to gain an interpretation of these measurable results as well as evaluate the performance of the framework an experiment was conducted in the private organization that provided the network dataset. The details and the design of the survey are first presented in detail. Moreover, the way in which the framework numeric results are combined with the survey responses to measure the framework efficiency is presented at the end.

5.4.1 Experiment Details

In order to evaluate the framework findings, the monitored employees input was needed to better understand their true actions on the network. For this reason, a survey was conducted on the period of 2 weeks in parallel with the network traffic monitoring. A subset of employees in the cyber security department of the private organization was willing to participate in a daily survey where they explained in measurable metrics their network behavior with regard to every mobile application they used on a daily basis. A total of 14 employees participated in the survey throughout the 2 weeks with a result of 70 survey submissions. An overview of the initial introduction text, user help texts as well as the survey questions are presented in Figure 5.6.

There is a good amount of work on how to design a survey in order to efficiently extract the information you need. Based on [59] and their suggestion on user behavior survey questions, an introduction text as well as clear instructions towards the survey participants facilitate the process and help to gather more accurate responses. Therefore, for each survey question there is a text to help the user better understand the requested information. Initially, the user is asked to provide the IP address of the mobile device he used as well as a unique private keyword. When it comes to large organizations, the IP address provided to a device can change during the period of the two week experiment because of the DHCP server configuration. To avoid attributing traffic to the wrong user and therefore introduce an error to our model, the IP address is requested alongside a unique keyword. In this way, IP addresses with the same secret keyword generated traffic belonging to the same user.

Moving forward, the user needs to define which applications he used throughout the day from a list of popular mobile apps as well as ones that he can manually submit. For each of the chosen mobile apps, short questions are asked regarding the mobile usage. In order to help the survey participants and gain more accurate responses, the application usage question is split in two sub questions with an ordinal scale. There are three available answers (Low, Medium, High) for the user to choose from. The participant is asked to express how much he used an application from a data usage and duration perspective. By giving two different perspectives, the user is able to better understand the concept of application usage and give a more accurate response. Although the ordinal scale is normally used for non

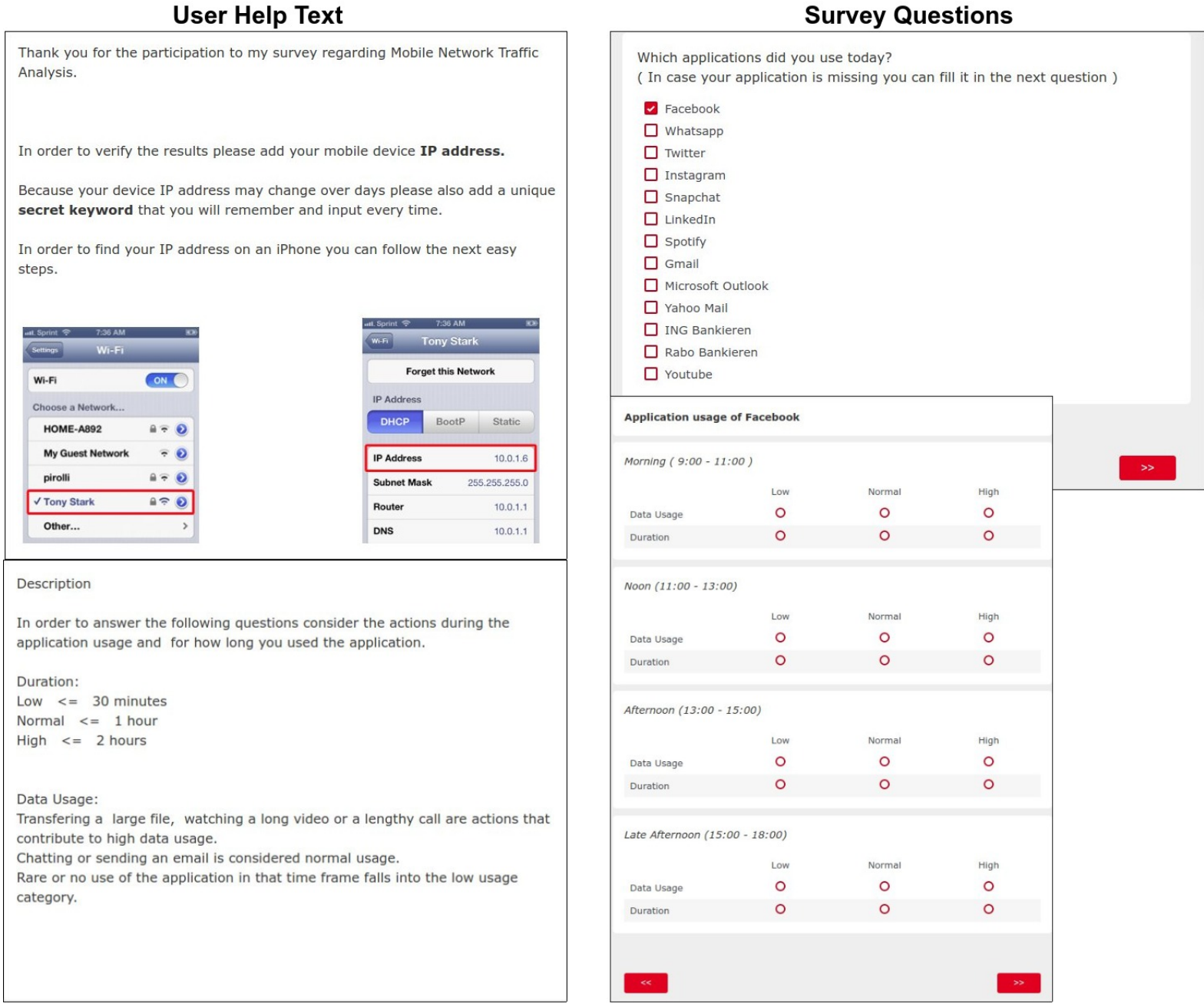

Figure 5.6: Survey Questions Overview

numeric concepts, a help text as mentioned before [59, 60] facilitates a user behavior question and gives a better understanding to the user. In this case, for the data usage perspective examples of actions that affect the data usage such as uploading a file or watching a long streaming video are provided to the user. For the time usage, the user is provided with some help information which matches Low to around 30 minutes, Medium to around 1 hour and High to around 2 hours of usage. Moreover as suggested by [61], response categories labeled with words give more reliable data than categories labeled with numbers. At the same time, a broad numerical range of options can be harder for the survey participant to choose from and accurately reflect his opinion. Of course, we can always assign numeric values later to make it easier to analyze responses.

Finally, the goal of the framework is not only to follow user behavior changes but also detect possible anomalies caused by user related attacks. For this purpose, anomalies were injected into the network by asking specific survey participants to greatly change their application usage randomly during one of the time frames and submit the corresponding action in the survey. Although the way each individual perceives an anomalous action is not the same, based on the participants feedback such actions relate to sending large files over an application repeatedly or using application functionalities that they normally avoid. This can be translated into scenarios of data exfiltration or device theft where we expect the user behavior to similarly greatly change. The decision to ask employees to change their behavior instead of manually injecting packets into the network was made so as have a more realistic scenario to evaluate.

5.4.2 Evaluation Approach

From every survey entry, the submitted answers describe the user application behavior from the user perspective on a qualitative scale. At the same time, the framework results provide a numerical evaluation of the change of user behavior between time frames from the network perspective. We define an approach on how to compare these two sources of information so as to evaluate the framework results.

First, the survey responses need to be translated from a range of Low, Medium, High to values that can be more easily compared against the framework output. However, the framework values express changes in behavior between the four time frames. In the same way, we introduce three transition labels that are based on the change of user answer between time frames. From the available values and for each time frame, the user can either express that his behavior remained the same and submit a value similar to the previous time frame or express that his behavior changes. When it comes to behavior change there are two scenarios. Either the user willingly changed his behavior or he is one of the survey participants who introduced an anomaly into the network. The three transition labels we present cover all these cases and are presented below. After parsing the survey answers so as to add the aforementioned labels a total of 150 stable transitions, 51 user dynamic transitions and 9 injected anomalies were recorded.

- **Stable Transition**: The submitted answer between the time frames remained the same

- **Dynamic Transition (user intended)**: The submitted answer between the time frames changed based on user intention
- **Dynamic Transition (user anomaly)**: The submitted answer between the time frames changed based on a user anomaly

Now that the survey answers have been labeled, we need an approach to compare the numerical framework output against these labels. The framework values are the cluster flow transitions and their goal is to reflect user actions and behavior change. The higher the value, the more dynamic the user behavior change is expected to be. Based on this concept, we turn this comparison into a classification problem of the transition values where the set of classification labels are the aforementioned survey labels. Based on their value, each cluster flow transition is going to be classified as one of the three available categories. The goal is for the classification to achieve the maximum accuracy which is the most correctly labeled transitions. For this purpose, we define two thresholds against which the transition values are compared in order to be placed into the categories. A visualization of the proposed approach is presented in Figure 5.7. If a cluster transition is below the low threshold then it is classified as a stable transition. When the cluster transition is between the two thresholds then it is a user intended transition and finally the anomalies which are expected to have the highest values are the ones above the high threshold.

Figure 5.7: Cluster Flow Threshold Labelling

The threshold values are distinct per application and shared by all users. The nature and category of mobile apps e.g. mailing, messaging, streaming heavily affects the size and duration of their network sessions and therefore the characteristics of the created clusters. As a result, the cluster transition values of all the users regarding each application are aggregated in a separate distribution so as to calculate the thresholds. In order to have unique thresholds for each user, a larger and longer dataset would be required so as to have a healthy value distribution. In order to identify the most suitable threshold values, the metric of quantile is used. Compared against other statistical metrics such as mean or standard deviation, the quantile is more resilient to outliers and non normal distributions and therefore can more efficiently split data points. Quantiles are cutpoints dividing a set of observations into subsets of equal size. The often used version of quantiles are the quartiles which separate a dataset into four subsets. However to achieve higher precision on the final result, the percentile metric is used which separates the dataset into a hundred equal subsets. As per every classification algorithm, the percentile value that matches a threshold is chosen when the highest classification accuracy is achieved. This translates into the highest number of

cluster transition values being correctly labeled based on their survey label counterparts. In the following chapter, the results of the experiment as well as an analysis of the employee application usage habits are described.

Chapter 6

Results

Initially in this chapter, we present a comparison between some known clustering algorithms so as to demonstrate the benefits of density based clustering that we described in the previous chapter.

Afterwards, the proposed framework output as well as the insight it provides are presented. First, the main goal of the thesis and the framework which is the ability to identify user behavior changes is evaluated. Moreover, the framework scalability on the total of the organization employees is demonstrated. By presenting the evaluation approach in the previous chapter, it is clear that the framework ability to follow user network behavior is measured through the accuracy of the classification problem of cluster transitions combined with the survey labels. Once the accuracy of the proposed framework is measured and its ability to reflect user behavior is established, a broad temporal and per application analysis is presented. The analysis identifies behavioral patterns from all the employees throughout the experiment as well as some insights about the differences between the employees behaviors. The statistical and cluster related information from the cluster transitions give important information regarding the behavior patterns and are the main source of information that will be used. Overall, the analysis of the framework output from all these different perspectives and utilizing all the available pieces of information provides a complete picture of the value and contributions of the framework.

Before diving into the aforementioned results, the list of applications involved in the analysis is presented in Table 6.1. The percentage of participation is estimated using the total number of sessions from each application. The list contains applications from three different categories messaging, streaming and mailing which enhances the diversity of the analysis. During the experiment of two weeks, various mobile apps were identified. However, not all of them were submitted by the users through the survey answers. Regarding the usage of the few applications not submitted, we do not have any information that reflect the user aspect and a possible way to verify the network aspect. Moreover, since privacy is an important aspect of the framework and all the users are anonymized we cannot obtain this information at a later stage. Finally, as mentioned before some employees were asked to greatly change their behavior and therefore inject anomalies into the network. The choice of applications to use while injecting the anomalies was solely chosen by the employee themselves. Based on the survey responses, the applications which include anomalous traffic are

Facebook, Gmail and WhatsApp.

Table 6.1: Mobile Apps under Analysis

Application Name	Participation in Total Traffic (%)
Facebook	30
WhatsApp	9
Telegram	7
Gmail	7
Outlook	9
YouTube	31
Spotify	7

6.1 Clustering Algorithm Comparison

In the previous chapter, we presented the three main requirements that density based clustering satisfies against the other type of clustering methods and why that makes it suitable for our thesis. In order to validate these arguments, we compare two widely used, in anomaly detection, algorithms against HDBSCAN. These two algorithms are the k-means and hierarchical clustering algorithm. One main disadvantage these two algorithms have against HDBSCAN is the requirement of providing a priori a number of clusters. There are a few established approaches [62, 63, 64] that help the user identify a good number of clusters although some of them introduce limitations that make them invalid for our research. The most well known method is the elbow one where a user can most of the times visually identify the optimal number of clusters based the amount of variance each new cluster introduces. Other approaches such as the silhouette score[65] or the gap score[66] provide a numeric value that can be used in an automatic way to determine a suitable number of clusters. In our thesis the clustering algorithm is applied a great number of times on the network dataset which makes a manual approach unrealistic to use. We tackle this problem, by choosing two different automatic methods to decide the optimal number of clusters for each algorithm.

For the k-means algorithm, the silhouette method [65] is used. The silhouette of a data point exhibits how close the point is to rest of the cluster members and how loosely it is matched to data points that belong to other clusters. An established method to choose the optimal number of clusters is to iterate through different cluster numbers and pick the one with the maximum average silhouette score for all the points. Since it is not practical to have a long range iteration, we take a different approach. Initially, the cluster numbers that will be tested are from 2 up to the number of clusters that HDBSCAN chose. If the optimal number of clusters is equal to the HDBSCAN one then the process is repeated with a increase step of 1 until the average silhouette score drops.

For the hierarchical clustering, there are two main approaches [63] on choosing the number of clusters. The first approach is simply choosing a static cut off level and based on the amount of links it cuts horizontally the number of clusters is chosen. In the second

approach, the representation of a dendogram is utilized alongside a specific metric provided by the algorithm called inconsistency. The second approach is chosen here because it is more flexible when it comes to choosing the number of clusters. The value of inconsistency compares the height of a link in the cluster hierarchy with the average height of the links below it. A link that connects two distinct clusters will have a high inconsistency metric. On the other hand, a link that connects clusters that are relatively close on the feature space will have a low inconsistency metric. A cutoff threshold for the inconsistency metric is chosen instead of the distance metric. The mean value of all the inconsistency metrics is used so as to detect the number of clusters.

Regarding the two input parameters of HDBSCAN described in the previous chapter, it's authors provide an understanding on how they affect the algorithm outcome. The minimum cluster size is a parameter that is closer to the dataset under analysis and should be a very small portion of its size. By providing a small percentage of the total data size the algorithm is not restricted to stopping when rather large clusters are found. At the same time based on the implementation of HDBSCAN, this does not mean that very small clusters are forced to be found since clusters are also bound by the threshold of the mutual reachibility distance. In our case and based on author suggestion, the value chosen is 5% of the size of the dataset each time the algorithm is applied. The k neighbour parameter plays a more dramatic effect on the clustering. The larger the value of k neighbours the more conservative the clustering will be since more points will be considered isolated and declared as noise. However, in our research the dataset is highly noisy and the algorithm is expected to be tolerant to this kind of points so the parameter is set to the minimum value of 2. The k neighbor value was chosen based on the author documentation on the suggested values and the notion of what it represents. When the value was increased, the algorithm declared as noise too many points of the dataset which was expected based on the dataset noisy nature. Overall, as we can see the HDBSCAN algorithm takes into consideration the concept of outliers and defines the clusters that are most flexible and suitable to represent the data points.

Now that the methods of choosing the number of clusters have been explained in detail, a comparison of the algorithms is presented. The three algorithms are compared based on the average number of clusters for each of the monitored applications and each time frame. In table 6.2, the average number of clusters per clustering algorithm and application are presented. The number of clusters have been rounded to the closest integer since a float number does not make sense with the notion of cluster number. As expected, the HDBSCAN algorithm has a higher number of clusters in all of the categories since as we mentioned in the previous chapter it is able to identify clusters of arbitrary shape. At the same time, the notion of outliers creates more distinct separations between clusters and therefore facilitate the separation of data points in clusters. Comparing k-means against hierarchical clustering, we can see that the second one is able to get closer to the number of clusters of HDBSCAN in some cases compared to the first one. Again, this can be attributed to the fact that hierarchical clustering is able to form arbitrary shape clusters where k-means can only create spherical shaped clusters. However, none of the algorithms is able to achieve the detailed clustering that HDBSCAN does.

Furthermore, a comparison based on the average cluster size for each of the clustering

Table 6.2: Average number of clusters per time frame

	Facebook	WhatsApp	Telegram	Gmail	Outlook	YouTube	Spotify
HDBSCAN	8	7	5	4	4	8	4
K-means	3	3	2	2	2	5	2
Hierarchical Clustering	5	4	4	2	3	6	2

Table 6.3: Average cluster size per time frame

	Facebook	WhatsApp	Telegram	Gmail	Outlook	YouTube	Spotify
HDBSCAN	16	13	12	25	20	10	20
K-means	60	36	35	56	47	27	43
Hierarchical Clustering	37	27	18	56	31	21	43

algorithms is provided. The results are presented in table 6.3. As we can see the size of the average cluster in HDBSCAN is considerably smaller than the cluster of k-means and hierarchical. This can be attributed to two main reasons. Firstly, HDBSCAN is able to provide more accurate and smaller clusters since it is more flexible when it comes to cluster shape. Secondly, an important aspect is the points that are included in each cluster per clustering method. As we highlighted in the previous chapter, algorithms such as k-means and hierarchical clustering do not include the concept of outliers. Therefore, inside these clusters are points that are forced to be part of a cluster even thought they are isolated from the set. On the other hand, HDBSCAN labels these points as outliers and they are excluded from the clustering process.

6.2 Framework Classification Results

The clustering process is applied on a dataset that consists of network sessions. The network sessions are considered as the reflection of user actions through the chosen 8 network features. This means that when a user changes his behavior, the network sessions during that time frame are expected to exhibit a change in values. As a result this translates to a correlated change in the cluster transition flow values which measures the changes in clusters of sessions. To better understand this chain of effects before moving on to the framework accuracy and classification of cluster flow values we present two examples. Based on the possible scenarios and the three different labels, an example visualization of two user dynamic cluster transitions (use intended and anomalous) is presented to get a better understanding on how exactly network sessions reflect user actions. Although the feature space has 8 dimensions, the network sessions are presented in two different pairs of features during the four time frames.

First, a case of a user intended dynamic transition is presented in Figure 6.1. The results in Figure 6.1 are presented in a log scale to better visualize the difference between each time frame sessions. In this example the WhatsApp usage of a random employee is presented. At his submitted survey entry, the user replied that his behavior was Medium during the time frames Morning, Noon and Low during the rest of the time frames. The highlighted subplots on the right refer to the time frames where the user expressed a change of behavior

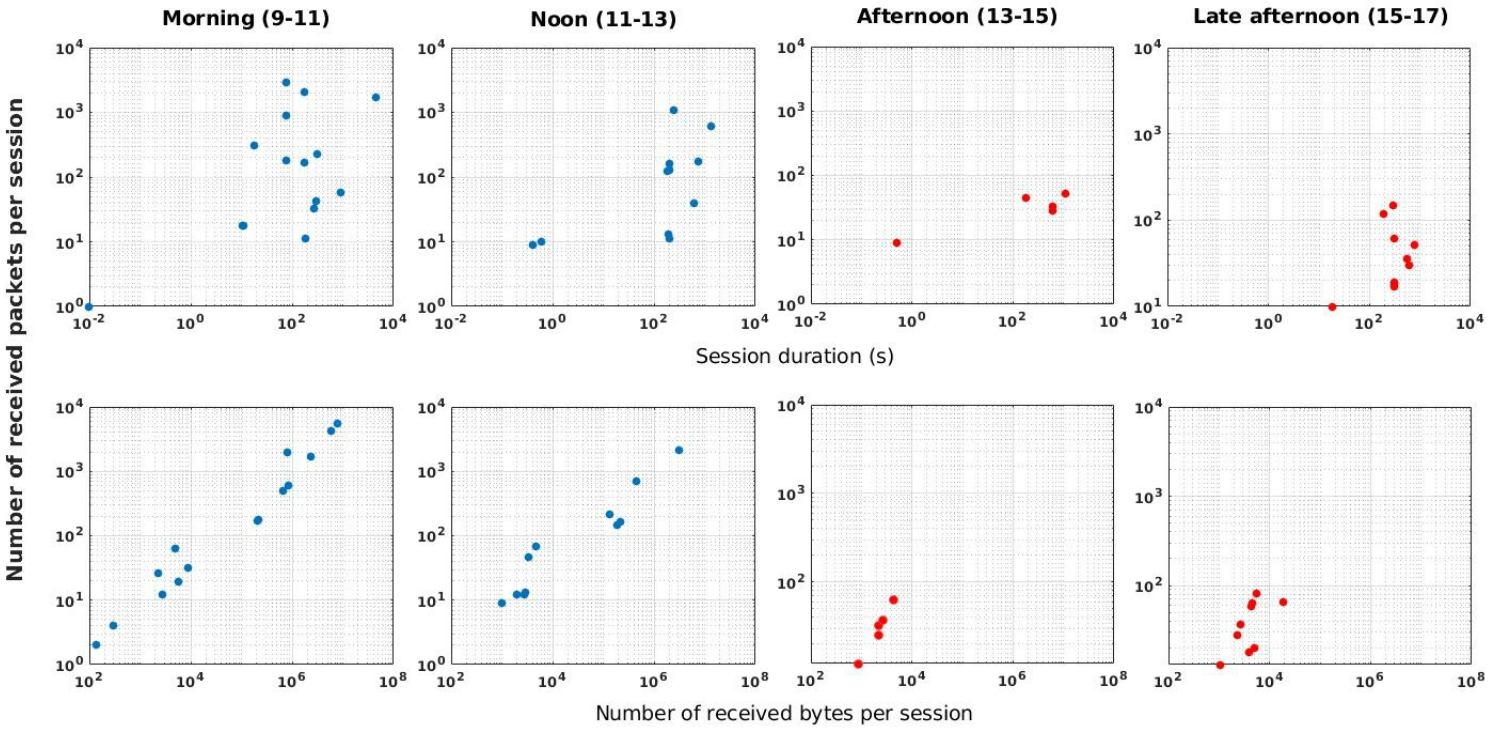

Figure 6.1: User Intended Behavior Change

compared to the two previous ones. Although the values on the session duration axis have the same value range, the values regarding the number of received packets are considerably lower. Moreover from the received bytes perspective, the two highlighted frameworks have distinct differences with the other two. Overall, as we can see the highlighted dataset points have a different value distribution compared to the previous two. At the same time, they exhibit a similar distribution in between them. From this example, we can better understand that when a user changes his behavior towards an applications there is a correlated change in the value distribution of the sessions which can be identified.

Moreover, an example of anomalous behavior on the Facebook application is presented in Figure 6.2. In this case, the user answered that he ”injected” the anomaly during the last time frame. Similarly as before, the sessions during the highlighted frame highly differ from the rest of the set. However in this case, the values appear to have an extremely higher diversity and range of values because of the fact that it is an anomaly instead of a simple behavior change. During the first three time frames, the sessions have a small duration and relative small number of packets. On the other hand, during the anomalous time frame session duration greatly increased as well as the number of sessions with higher amount of traffic. Again, the network sessions seem to exhibit changes that align with the changes in the user behavior and the degree of change in the session values also seems to align with the magnitude of behavioral change.

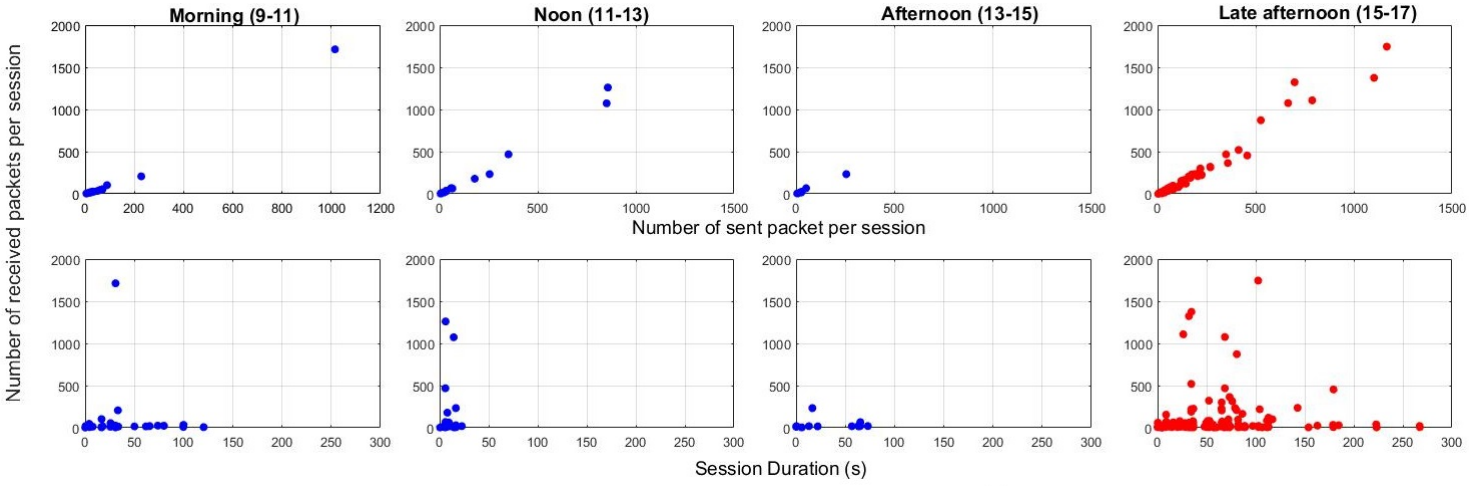

Figure 6.2: User Anomalous Behavior Change

Although these two examples are a small sample of the numerous similar cases in our research, they verify that sessions indeed exhibit a change in values when there is a user behavior change. More importantly, the change in session values is correlated with the change in behavior since we saw that an anomalous change in behavior has a more drastic effect on the session values. In the next section, we evaluate how these changes in sessions and as a result on the framework output can accurately identify user behavior changes based on user input.

6.2.1 Framework Classification Accuracy

As described in the previous chapter, the ability of the framework to follow user behavior changes and reflect them through its numeric output is measured with the help of the conducted survey. The three distinct labels added to each survey response are utilized as a ground truth to be compared against the labels added to the cluster flow transition. Each cluster flow transition is labeled based on the comparison of its value against the two threshold we introduced. Based on these concepts, the accuracy of the framework is equal to the accuracy of the classification problem and the two thresholds are the two configuration parameters that can maximize the accuracy. Based on the analysis of the three clustering algorithms, the accuracy results of HDBSCAN, hierarchical clustering and k-means are presented in this section.

The specific values of the two parameters are decided based on the value that provides the highest framework accuracy and minimum false positive rate accordingly. As mentioned in the previous chapter, the cluster flow transitions are separated based on the application that the sessions originated from. With the same reasoning, a different threshold is applied on each of the application distribution. However regarding the high threshold that classifies transitions as anomalous, there is a small set of observations that fall into this category and three distinct applications. Moreover, the percentile metric is based on the dataset

value range and therefore introducing a high threshold for applications that do not even contain anomalous instances would not provide any meaningful output. Therefore, the high threshold refers only to the three applications where users exhibited anomalous behavior and after experimenting with different values, the optimal threshold is at **95%** percentile value.

On the other hand, the low threshold which differentiates stable from user intended transitions highly differs between applications. This occurs because all the applications include instances of both stable and user intended transitions and they exhibit a different network behavior. In order to identify the optimal low threshold, its value is continuously increased as long as the accuracy rate changes accordingly. As expected, after a certain value of low threshold the accuracy rate starts to drop. This means that the framework has adopted a very strict labeling behavior and cases of user dynamic transitions are mislabeled as stable ones. In Table 6.4, the average low threshold shared by all applications alongside the accuracy rate are presented for the three clustering algorithms. For hierarchical clustering the optimal threshold is at **75%** percentile with an accuracy of **61.9%**, for HDBSCAN the optimal threshold is at **85.5%** percentile with an accuracy of **95.2%** and for k-means the optimal threshold is at **75%** percentile with an accuracy of **57.61%** . As we can see the hierarchical clustering algorithm and k-means have a significantly lower accuracy than HDBSCAN as well as a lower low threshold. The low accuracy is mainly caused by the fact that both algorithms aggregate sessions into more generic and larger clusters that include outliers. As we explained before, forcing outliers to be parts of the final clusters leads to extreme values inside each cluster. These extreme values lead to high spikes in the pairwise comparison results which are key parts of our cluster comparison algorithm. This translates to cluster flow transitions whose values are, because of the inclusion of outliers, not correlated with the magnitude of a user behavior change and therefore lead to a high number of misclassifications. This phenomenon is expressed both in the low accuracy of the clustering algorithms as well as the the low percentile value used as a threshold. Moreover, the k-means algorithm has a lower overall accuracy than hierarchical clustering which comes in agreement with the previous results of average cluster size and number.

Table 6.4: Clustering Algorithm Classification Accuracy

Average Low Threshold (%)	HDBSCAN Accuracy Rate (%)	Hierar. Clustering Accuracy Rate (%)	K-means Accuracy Rate (%)
65	78	52.3	50.47
70	83	54.7	52.8
75	87.1	**61.9**	**57.61**
80	91.9	59.5	56.66
85.5	**95.2**	57.1	55.23
88	92.8	51.9	50.95
90	90.47	50.4	49.52

In order to get a more detailed view on the threshold for each applications as well as how the hit and misses of the framework are distributed we present the specific threshold value for each application in Table 6.6. The detailed accuracy of the model is presented in Table 6.5. Both of the tables refer to the results derived from HDBSCAN clustering since it is the optimal clustering method. As we can see the majority of cases where the framework misclassifies is when the survey participant answered that his behavior remained stable

between two time frames but the model identified changes in his cluster flow above the low threshold. Factors that explain these scenarios can be attributed to the introduction of human error where the survey participant misjudges his application usage behavior. However, this type of cases are acceptable since they are part of an evaluation implementation that includes a survey. On the other hand, the misclassification of user intended transitions are related to the ability of the model to distinguish different type of transitions.

Table 6.5: Detailed Framework Accuracy

Transition Label	Accuracy (%)
Stable	94.6 (142/150)
User Intended	96.0 (49/51)
User Anomaly	100 (9/9)

Table 6.6: Application Percentile Values

Application Name	Low Threshold(%)
Facebook	85
WhatsApp	87
Telegram	84
Gmail	88
Outlook	85
YouTube	85
Spotify	85

Combining the cluster algorithm comparison purely from a cluster characteristic perspective with the accuracy results in table 6.4, we can establish that the density based clustering algorithm is more suitable for the purpose of clustering noisy network traffic and modeling user network behavior. Therefore, for the rest of the section all the presented results will based only on the output of HDBSCAN clustering. Furthermore, the results from the combined output of the framework and the survey verify the thesis goal of creating a framework that is able to closely follow and reflect user behavior changes. The framework is able to follow user behavior with high accuracy both between stable transitions as well as dynamic ones. Moreover, the notion of cluster flow values is able to grasp the user behavior that is reflected on the network. The value range is closely aligned with the actual human actions where a dynamic change of behavior is directly translated to a high value. This is highlighted by the model ability to identify anomalies perfectly as well as by the fact that anomalous changes are the highest values in their corresponding distribution.

6.2.2 Framework Scalability

The survey answers provide the ground truth regarding the user behavior during the monitoring phase and are therefore the main source of information to evaluate the framework. However, the network data generated by the company employees that did not participate in the survey can be still utilized. The information can be used to evaluate how representative is the subset of survey participants against the total company staff. More specifically, this will highlight whether the framework is able to reflect properly the user actions on a larger scale. To achieve this, the same methodology as before is applied on the overall network traffic. This results in a cluster transition distribution with far more observations for each application. To give a better understanding, the average number of total users per application is presented in Table 6.7.

Table 6.7: Company Wide Application Users

Application Name	Average Users per Day
Facebook	967.2
WhatsApp	986.0
YouTube	697.2
Gmail	247.4
Spotify	107.1
Outlook	350.6
Telegram	11.5

Table 6.8: Application Percentile Values

Application Name	Low Threshold(%)
Facebook	82
WhatsApp	82
Telegram	84
Gmail	80
Outlook	79
YouTube	85
Spotify	87

Table 6.9: Framework Accuracy

Transition Label	Accuracy
Stable	92.6 (139/150)
User Intended	88.2 (45/51)
User Anomaly	100 (9/9)

Based on the previous section, HDBSCAN is far more accurate than the hierarchical clustering and in this section the results are presented based on HDBSCAN only. Using the same evaluation process as before, the framework has a total of **91.9%** accuracy rate with an average of **82.7%** lower threshold and **97%** high one. The detailed accuracy of the model is presented in Table 6.9. Similarly as before, the high threshold is shared by all three applications that contain malicious transitions. The detailed low threshold of each application is presented in Table 6.8. As we can see, the new low threshold values for each application have changed compared to the ones that included only the survey participants. These changes on the low threshold for each application can be interpreted in two ways based on the fact that the behavior of a big amount of users was added. In the case that more users who exhibit a rather stable behavior are introduced to the distribution, the percentile value is expected to go higher so as to provide a more accurate threshold. On the other hand, when users that exhibit a rather dynamic behavior are introduced then the percentage of people in this category increases. Therefore, a lower percentile threshold is needed to represent them.

Overall, the framework key purpose which is to be able to closely monitor the user network behavior and identify changes is again established. This means that the survey participants are a good representation of the total set of users and their actions are a good representation of actions we expect to find in the organization network. The accuracy rate drop is attributed mostly to the fact that newly introduced behavior shifts from the rest of the employees do not contain survey labels. Therefore, the framework does not have a ground truth to compare these new introduced values which would further increase the framework accuracy on the large scale.

An important factor is that the framework is still able to identify all the anomalies and a very good amount of user intended behavior changes. This means that the value range of

cluster transitions used on the survey scope is a good representation of the value range on a larger scale. Moreover, user anomalies are considerably high even after the introduction of a large amount of users. This proves that the proposed cluster flow approach is able to grasp accurately the notion of user actions and reflect behavior change through its numeric value.

In the next section, we try to extract behavioral patterns and behavior trends between users. Since there are no contextual information regarding the users such as gender, job title so as to utilize for the behavior patterns we focus on the temporal and per-application aspect. Moreover, we provide a comparison analysis on the daily user application behavior. We try to identify potential clusters on the user daily behavior which would yield groups of employees that use a specific application in the same manner or applications where users behaviors highly differ.

6.3 Framework Behavioral Patterns

Initially, the framework ability to identify behavior changes was established by utilizing the survey responses and considering the survey participants as the set of users. Moving forward, the framework scalability was measured so as to verify that the sample extracted from the organization population was representative and that the model is able to scale even when a considerably larger set of cluster flow transitions is introduced. With these two key points proved, the main research question of the thesis is satisfied. The framework is a user network behavior model on an application level that reflects behavior changes, detect anomalies and exhibits the ability to scale.

Moving forward, we evaluate the framework ability to identify behavioral patterns shared by users as well as clusters that denote similar user behavior. The cluster flow transition values can be utilized to identify and suggest behavioral patterns among employees from an application perspective as well as a temporal one. In this section, we initially present the behavior trends of the average employee throughout the two weeks for all the time frames as well as all the applications. Afterwards, we try to compare the behavioral habits of employees in between them and present a daily cluster flow comparison. Moreover, the values of the daily cluster flow comparisons are visualized through a dendogram so as to identify users with similar behaviors. Similarly as before, since the frameworks ability to scale has been proved the following analysis is performed on the total of the private organization employees.

6.3.1 Employee Behavioral Patterns

Each cluster flow transition value denotes the change in cluster stability and cluster movement in the feature space. The change in actions and overall behavior of a user towards a specific application between time frames is directly translated into a change in cluster flow. In order to introduce a temporal aspect on the analysis, we aggregate these cluster flows into groups based on the day they were generated, the specific time frames they refer to as well as the application that they belong to. More specifically, the repeating days of each week of the experiment are split in different groups so as to further analyze behavioral patterns on a daily level. By applying these separations on the values, we are able to identify changes in

behavior for each one of the 10 days of the experiment as well as each one of the three available time frame transitions. Finally, the mean value of cluster flow transition of each group is computed. Therefore, these values represent the average change in behavior between all time frames for each day of the experiment and each distinct application.

With the notion of a cluster flow transition in mind, an overall low mean value expresses a stable behavior and a relatively high one means that the average user greatly changes his behavior regarding the specific application. Moreover, the changes in cluster flow values between time frames provide insight on specific behavior patterns and changes in behavior. A relatively low value spread between time frames means that the user exhibits a stable behavior throughout time without any major changes. On the other hand, a relatively high value spread between time frames means that the user continuously changed his behavior during those time frames. Finally, a spike in the cluster flow values would mean that the user changed his behavior once but then retained the same type of behavior.

In order to better understand the following figures, the x-axis labels denote the comparison between consecutive time frames. The label M-N stands for Morning-Noon, N-A stands for Noon-Afternoon and A-L stands for Afternoon-Late Afternoon. For a better visual representation, the first three applications belonging to the messaging category are presented in Figure 6.3 and the rest four applications are presented in Figure 6.4.

In both Figures 6.3 and 6.4, the cluster flows for each application have the same repeating patterns throughout the days with a few exceptions. This means that the users do not deviate from their normal behavior routine and use their mobile applications in the same manner in a repeating fashion. As we can see, this leads to cluster flow transition values that are within the same range throughout each day and closely follow each other. By taking a closer look on repeating days, we can identify that same days in week 1 and week 2 seem to share a more distinct pattern in between them. However, this hypothesis cannot be verified with certainty since there are cases where the behavior between repeating days is quite different such as WhatsApp on Tuesday and Wednesday. In order to strengthen this assumption, more repeating days and therefore an experiment of longer duration are required.

Furthermore, we can identify specific patterns on a per application level. In Figure 6.3, Facebook has a distinct pattern of relatively high values between the first two time frame shifts and then a drop in the cluster flow value. This can be attributed to the fact that most employees increase the usage of Facebook during and after lunch and therefore cause this value pattern starting from the morning time frame up to the afternoon. Moreover, the Facebook cluster flows on Thursday and Friday in both weeks have smaller values than the rest of the week. This means that the users started to exhibit a more stable, either high or low, behavior during those days and thus cause a smaller cluster flow transition. Regarding the other two messaging applications both of them have relatively small cluster flow values which means that user exhibit a stable behavior throughout the day with a few deviations. Regarding Figure 6.4, the applications do not seem to exhibit a distinct repeating pattern on a time frame level apart from the overall pattern that we mentioned before on a day level. YouTube stands out from the rest and has a comparatively high cluster flow value throughout the two weeks which means that users change highly their application usage throughout the day.

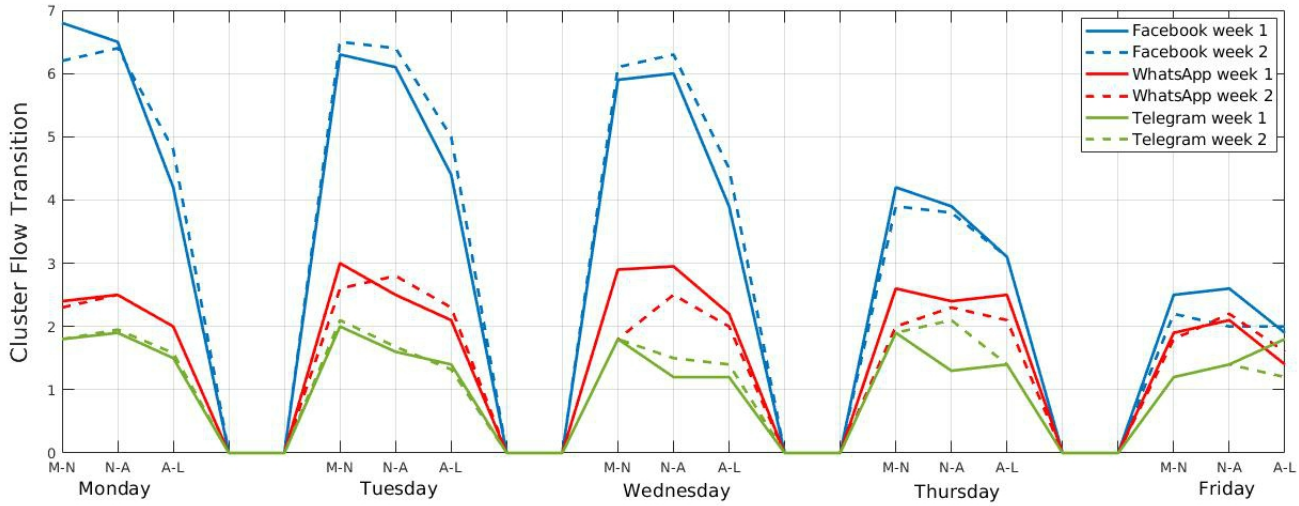

Figure 6.3: User Behavior Detailed Patterns

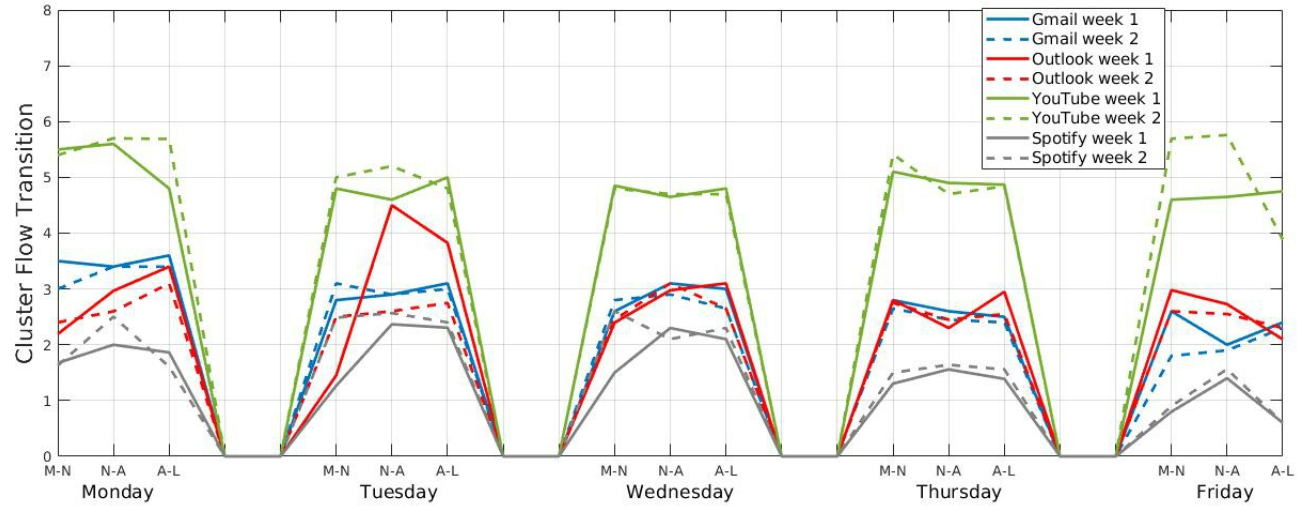

Figure 6.4: User Behavior Detailed Patterns

6.3.2 Employee Behavior Comparison

The previous analysis is focused on behavior patterns and habits based on the type of application used but with a per user approach. The cluster transition values are calculated by comparing sessions and clusters generated each time by a single user and then computing the mean value. In order to get a better understanding on the inter relationship between user behavior patterns, a direct comparison is applied. Because of the anonymity of the network sessions, there are no contextual information such as job title or age based on which an analysis can be performed. However, the difference in user behavior from a temporal perspective can be analyzed.

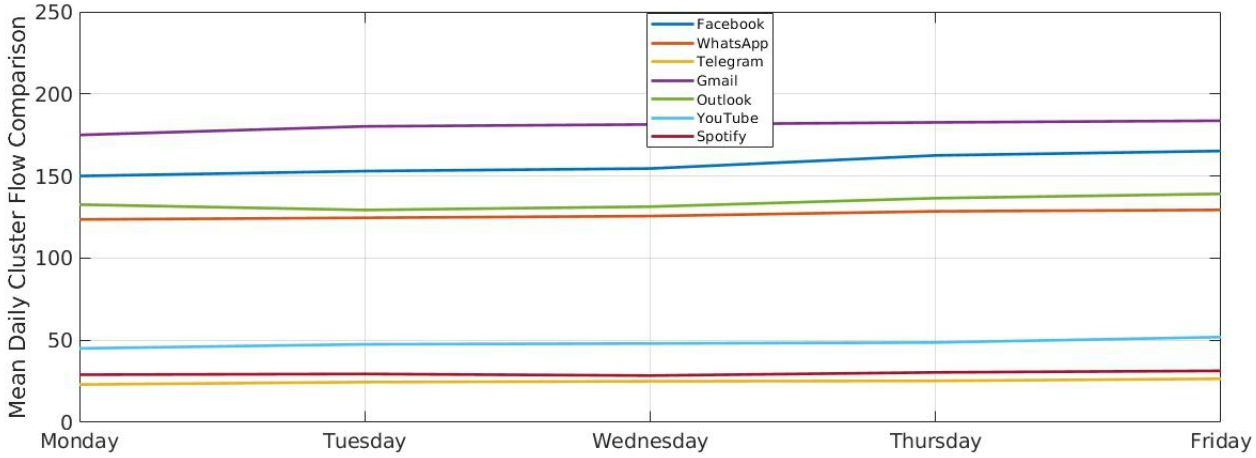

Figure 6.5: Daily User Flow Comparison per application

To achieve this, the network sessions of each user are aggregated on a daily level and HDBSCAN is applied on the daily set. This leads to a set of clusters that reflects the user behavior throughout the whole day. Since we want to compare the behavior between users, the cluster comparison algorithm presented in the previous chapter is applied between the daily clusters of all users. This results to a set of values that denote how different in size and position are the daily clusters of users. Therefore these values can be considered as the daily cluster flow comparison between all the users. In Figure 6.5, the mean value of all the cluster flow comparisons are presented for each day of the experiment. A relatively high mean value implies that users have a rather different behavior regarding that application in between them. On the other hand a low mean value implies that the daily clusters of the users present similarities in size and position in the feature space and therefore so does their behavior. The applications create two distinct groups based the range of values of cluster flow comparison. Facebook, WhatsApp, Gmail and Outlook exhibit a high value and therefore user behavior when it comes to these applications highly differs in between them. On the other hand, users seem to have a very similar behavior when it comes to Spotify, Telegram and YouTube. Moreover, the range of values in Figure 6.5 remain rather stable throughout the week with a few increases towards the end of it. This comes in agreement

with the previous results on the mean user behavior and the repetitiveness of behavior patterns. Since the per user behavior patterns seem to be rather stable throughout the days, a comparison between the users should exhibit the same type of stability.

Furthermore, to better comprehend the results in Figure 6.5, we pay a closer look on the daily cluster flow comparison between users and how these values cluster together and yield possible groups of users with similar behavior. Based on Figure 6.5, we choose two applications, Facebook and YouTube, from the two distinct groups to evaluate how similar or different are employee behaviors and if they correspond to the mean values we previously measured. From the aforementioned process of comparing daily user cluster behaviors, we keep all the comparison values between users for each day and each application. Instead of measuring the mean this time, we use this matrix of daily user comparisons so as identify users who exhibit a similar daily behavior. To achieve this, we take advantage of the concepts of dendograms and hierarchical clustering.

In hierarchical clustering, the algorithm is provided with a distance matrix between all points and starts by grouping points that are very closely together in sub clusters moving up to a singular cluster that contains all the data points. Dendograms are able to visualize this process and exhibit similarities and possible clustering trends between data points when provided with the distance matrix. Based on the hierarchical clustering process, the top of dendograms represent all the data points as one singular cluster and start to separate them into groups as they move downwards. The separation is based on the distance between points and the length of each separation denotes the number of data points split and therefore potential clusters. When a dataset includes points which are close together and should be clustered then the dendogram is expected to have even separations. On the other hand, a dataset with points that are spread throughout the feature space and cannot be easily clustered is expected to have long and uneven separations.

In our thesis, the matrix that holds all user daily cluster flow comparisons among all the employees is provided to the dendogram. Therefore, the potential cluster patterns will translate to users having a similar daily behavior among them. On the other hand, the lack of cluster patterns or isolated points corresponds to users having a different daily behavior. We pick the two most busy days for each of the two applications where busy translates into a high number of employees using the applications that day. The reason why the two most busy days are presented is to further validate the patterns found on the histograms and to have a relatively high number of users to analyze. In Figure 6.8, the dendograms of the daily cluster flows of users for YouTube are presented. As we can see in both sub Figures, there seem to be groups of users who exhibit similar daily behavior and are therefore grouped together. Of course, there are certain users who exhibit a highly different behavior from the rest but this is expected in a large number of observations. However, as the dendograms starts to move to points that are closer to each other, there are distinct clustering patterns between the points. Finally, as the separations becomes more strict the number of points separated every time start to gradually decline denoted by the green separation lines. This can be translated as a relatively stable cluster being further separated because of the algorithm process. This insight about users having similar behavior and forming potential clusters can be extracted from both of the different days in Figure 6.6 and Figure 6.7.

Moving forward, Figure 6.11 presents considerably different dendograms. In Figure 6.9

and Figure 6.10, we have the dendograms for each of the two most busy days of Facebook usage. As we can see, the patterns of data points being close to each other are more rare. On the other hand, the dendogram exhibits multiple points that are isolated from the group since it includes uneven separations with a large number of points at the beginning. Moreover, the majority of separations occur as the dendogram moves towards its bottom which can be interpreted as a lack of potential clusters with a relatively high number of members. This phenomena is observed in both of the unique days of facebook usage.

Overall, the dendograms express similarities and differences between the user daily behaviors as well as help us to identify potential clusters created by users. This provides insight on the level of similarity on the user behavior based on the application on a daily level. The insight provided by dendograms comes in agreement with the results extracted in the first part of the user behavior comparison. Although, there are a few cases of users either exhibiting a very different behavior from the rest or users having a similar behavior in both applications the overall mean value range corresponds to the dendogram results. The results in this section provide further validation and value to the cluster flow algorithm we proposed earlier. By utilizing this numeric expression of user behavior, we are able to identify users with similar behavior as well as behavioral patterns regarding specific applications throughout the day.

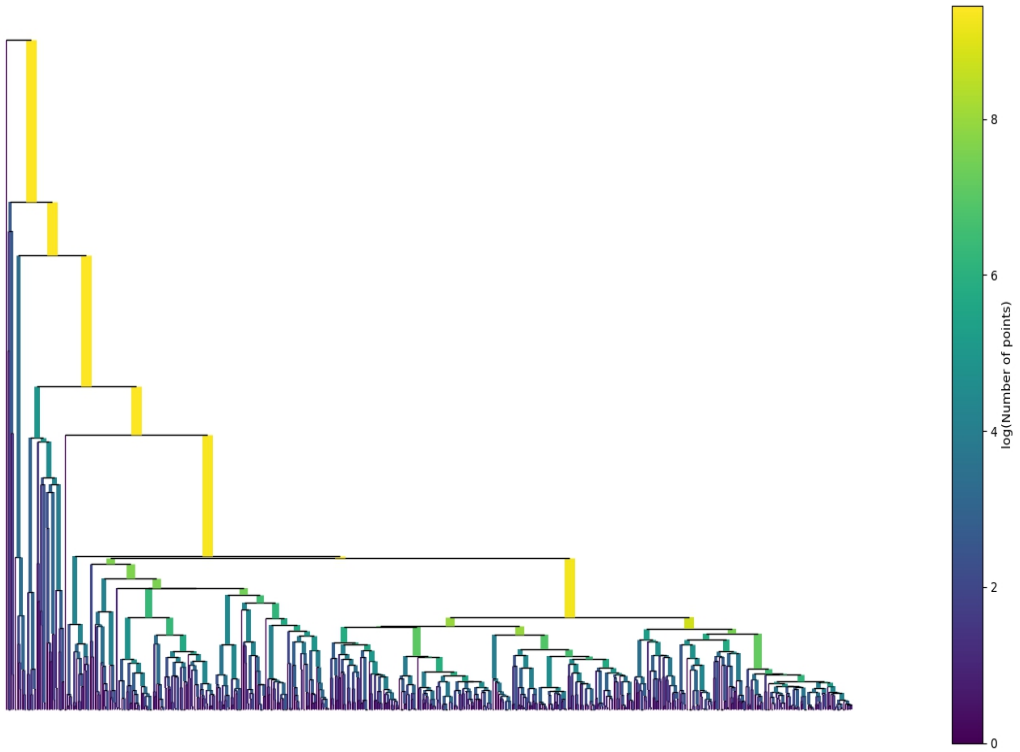

Figure 6.6: Cluster Flow Dendogram Day 1

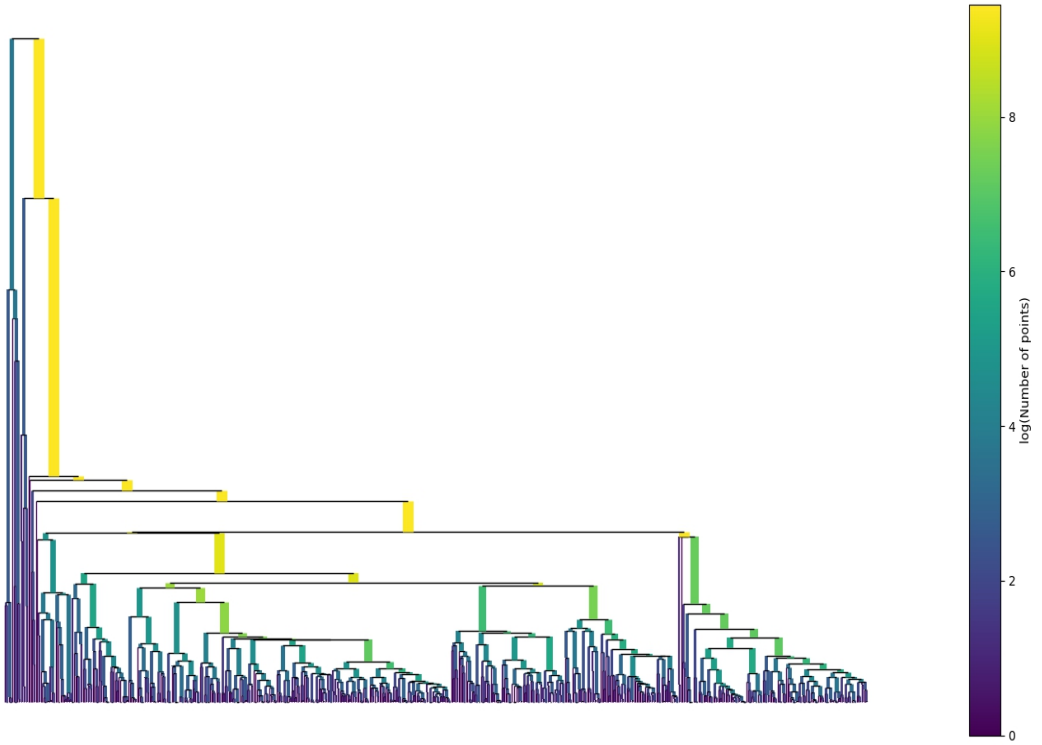

Figure 6.7: Cluster Flow Dendogram Day 2

Figure 6.8: YouTube User Daily Cluster Flow Comparison

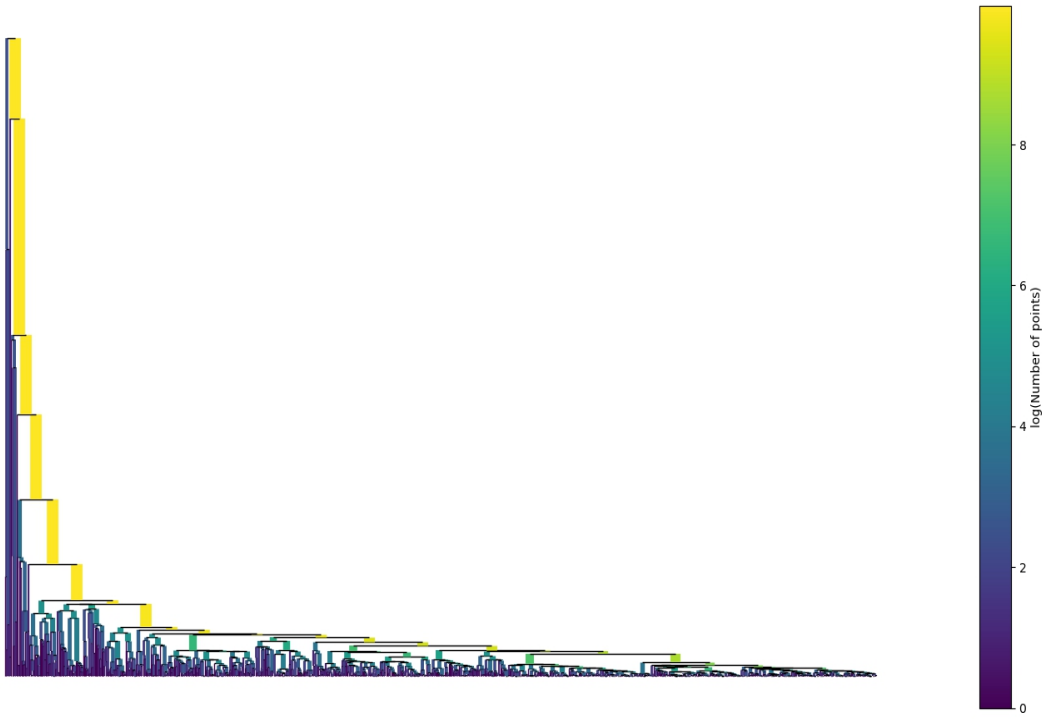

Figure 6.9: Cluster Flow Dendogram Day 1

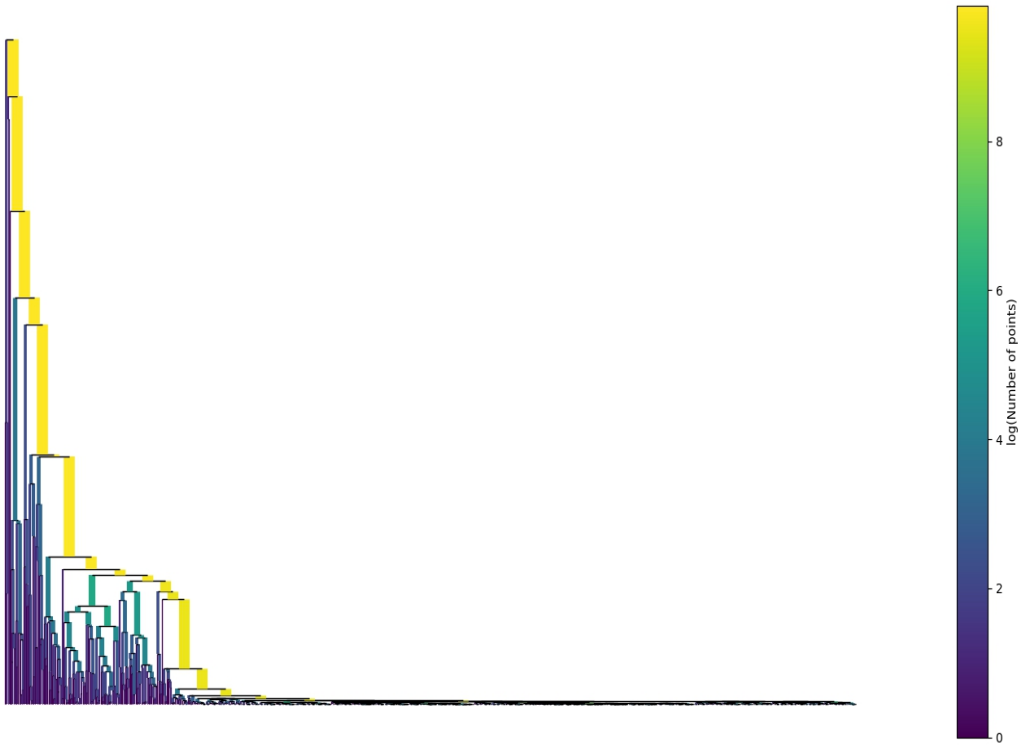

Figure 6.10: Cluster Flow Dendogram Day 2

Figure 6.11: Facebook User Daily Cluster Flow Comparison

Chapter 7

Conclusions and Future Work

Nowadays, organization networks are facing an increased number of different attacks and existing intrusion and anomaly detection systems fail to keep up. The landscape of attacks is becoming more broad from exclusive technical attacks to user related and social engineering ones. Moreover, as more and more organizations allow employees to freely connect to the company network with personal devices a closer look to the network traffic they generate needs to be paid. At the same time user privacy issues raise concerns for approaches that require employees to install company applications on their device.

For these reasons, the focus of the thesis was the modeling and analysis of user network behavior on a mobile application level. The proposed framework has a passive monitoring approach and tries to utilize the collected network traffic purely based on the information included in the packet headers. The goal of the framework is to be able to closely monitor and detect user behavior changes as well as anomalies. This type of changes can be attributed to both normal user behavior as well as possible attack scenarios such as data exfiltration or device theft.

In the context of the research questions stated at the beginning of the thesis, the combination of the detailed related work analysis and the presented results is able to answer all of them and present explanation and limitation regarding some of the answers.

Firstly, the transformation of raw network traffic into meaningful sessions and the further separation based on the mobile application traffic is able to provide us with a useful dataset for the purpose of the thesis. Based on the related work on application identification, the payload approach was chosen and more specifically a non intrusive scalable header-only approach. After analyzing the protocols found on the organization network as well as the documented protocol communications we identified key elements that contribute to the extraction of the application that owns the network traffic. With the proposed novel algorithm and the utilization of string similarity algorithms, we are able to identify all the mobile applications mentioned by the users through the survey. Moreover, the amount of traffic attributed to each application comes in agreement with the survey responses about application usage which gives us the confidence that our algorithm has a minimum amount of misclassifications.

Moving forward, an analysis of the existing work on user behavior anomaly detection exposes the lack of robust and user oriented solutions that can keep up with the dynamic

nature of user behavior. After analyzing the weaknesses of existing work and techniques that have yet to be exploited, we decided to utilize clustering algorithms to tackle this task. More specifically, the benefits of density clustering against other algorithms are presented and measured in the context of the thesis. Afterwards, the proposed framework is able to translate user actions and behavior throughout the day to a numeric estimation. This metric expresses a temporal overview of the use behavior changes with the help of our novel cluster flow algorithm. To evaluate and validate the proposed cluster flow algorithm and overall framework, we conducted an experiment so as to obtain the user behavior from the actual user perspective. The framework achieved great accuracy when it comes to user behavior changes as well as anomalies on the network. Furthermore, the framework ability to scale on the whole organization is validated as well as the fact that the extract survey participants are a good representation of the whole organization.

Finally, the cluster flow output is utilized in order to extract possible behavioral patterns as well as compare user behaviors on a daily level. The model is able to identify distinct repeating behavior of the employees. This means that people tend to have a stable behavior everyday with a few exceptions as well as specific behavioral patterns when it comes to specific applications. Furthermore, a daily comparison of the user behavior is presented. Although the comparison is limited by the lack of contextual information, we are able to identify distinct groups of users that share the same behavior as well as an overview of user behavior similarity depending on the applications.

When it comes to future work, the modularity of the proposed framework gives a lot of freedom on how the cluster algorithm results can be utilized. The framework results can be used to perform a more social oriented analysis on behavior patterns and application usage. However, this would require a survey that asks users more personal questions regarding their gender, job title, personal preferences etc. This can raise some concerns from a privacy perspective and user may not be willing to participate. Moreover, the scalability evaluation of the framework gives motivation for future work that can utilize a per person cluster transition distribution instead of an application wide one. However, to achieve stable results with this approach there are certain challenges that must be addressed. The most important problem is the ability to scale up a survey inside a large organization which can be a rather difficult task.

Overall, the proposed approach on clustering with the use of density based clustering as well as the concept of cluster flows is a promising concept on the section of clustering on user network behavior modeling. The metric of cluster flow seems to be able to closely follow user actions and provide the means of a user behavior comparison that has not been proposed yet. After tackling various problems and limitations presented in related work, the propose framework efficiency is verified with the usage of real network traffic. Moreover, the presented insight on behavior patterns as well as the accuracy of the framework on following user behavior verify the proposed concepts and algorithms. Finally, the framework results regarding scaling give motivation and potential exploitation for future work.

Bibliography

[1] Number of mobile-only internet users now exceeds desktop-only in the u.s. https://www.comscore.com/Insights/Blog/Number-of-Mobile-Only-Internet-Users-Now-Exceeds-Desktop-Only-in-the-U.S, 2015.

[2] SANS Institute. Sans mobility/byod security survey. Report, 2012.

[3] T. Shumate and M. Ketel. Bring your own device: Benefits, risks and control techniques. In *IEEE SOUTHEASTCON 2014*, pages 1–6.

[4] Mobile Marketing Statistics compilation. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/, 2017.

[5] McAffe. Mobile threats report: Whats on the horizon for 2016. Report, 2016.

[6] Roman Unuchek. Mobile malware evolution 2016. Report, Kaspersky Lab, 2016.

[7] MobileIron. Mobile security and risk review, third edition. Report, MobileIRon, 2016.

[8] A. J. Burns and M. E. Johnson. Securing health information. *IT Professional*, 17(1):23–29, 2015.

[9] Medicine under fire: how to hack a hospital. https://usblog.kaspersky.com/hacked-hospital/6685/, 2016.

[10] Benenson Zinaida. Exploiting curiosity and context: How to make people click on a dangerous link despite their security awareness. In *Black Hat*.

[11] Murugiah P. Souppaya and Karen A. Scarfone. Guidelines for managing the security of mobile devices in the enterprise. *NIST*, 2013.

[12] Y. Wang, J. Wei, and K. Vangury. Bring your own device security issues and challenges. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 80–85.

[13] Keunwoo Rhee, Sun-Ki Eun, Mi-Ri Joo, Jihoon Jeong, and Dongho Won. *High-Level Design for a Secure Mobile Device Management System*, pages 348–356. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[14] Jaeho Lee, Yongjin Lee, and Seung-Cheon Kim. *A White-List Based Security Architecture (WLSA) for the Safe Mobile Office in the BYOD Era*, pages 860–865. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[15] Alessandro Armando, Gabriele Costa, and Alessio Merlo. Bring your own device, securely. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1852–1858, 2480707. ACM.

[16] P. Kodeswaran, V. Nandakumar, S. Kapoor, P. Kamaraju, A. Joshi, and S. Mukherjea. Securing enterprise data on smartphones using run time information flow control. In *2012 IEEE 13th International Conference on Mobile Data Management*, pages 300–305.

[17] Oyindamola Oluwatimi and Elisa Bertino. An application restriction system for bring-your-own-device scenarios. In *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, pages 25–36, 2914645. ACM.

[18] S. Chung, S. Chung, T. Escrig, Y. Bai, and B. Endicott-Popovsky. 2tac: Distributed access control architecture for bring your own device security. In *2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom)*, pages 123–126.

[19] G. Costantino, F. Martinelli, A. Saracino, and D. Sgandurra. Towards enforcing on-the-fly policies in byod environments. In *2013 9th International Conference on Information Assurance and Security (IAS)*, pages 61–65.

[20] Eun Byol Koh, Joohyung Oh, and Chaete Im. A study on security threats and dynamic access control technology for byod, smart-work environment. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2014.

[21] Security intelligence, Sense Analytics for protecting assets, and information from advanced threats. `http://www-03.ibm.com/software/products/en/qradar-siem/`.

[22] Splunk. `https://www.splunk.com/`.

[23] B. Zeng, D. Zhang, W. Li, G. Xie, and G. Zhang. Design and implementation of a network behavior analysis-oriented ip network measurement system. In *2008 The 9th International Conference for Young Computer Scientists*, pages 374–379.

[24] NetFlow. `http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html`.

[25] H. Hou, T. Xu, and R. Zou. Design and implementation of ip network behaviour analysis system based on netflow. In *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 533–538.

[26] Q. Xu, Y. Liao, S. Miskovic, Z. M. Mao, M. Baldi, A. Nucci, and T. Andrews. Automatic generation of mobile app signatures from traffic observations. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1481–1489.

[27] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song. Networkprofiler: Towards automatic fingerprinting of android apps. In *2013 Proceedings IEEE INFOCOM*, pages 809–817.

[28] monkeyrunner. `https://developer.android.com/studio/test/monkeyrunner/index.html`.

[29] A. Zquete and M. Rocha. Identification of source applications for enhanced traffic analysis and anomaly detection. In *2012 IEEE International Conference on Communications (ICC)*, pages 6694–6698.

[30] Y. Kumano, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka. Enhancing immediacy of identification with multi-stage application identification. In *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–2.

[31] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil. Application identification via network traffic classification. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 843–848.

[32] Y. J. Won, Park Byung-Chul, Ju Hong-Taek, Kim Myung-Sup, and J. W. Hong. A hybrid approach for accurate application traffic identification. In *2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, pages 1–8.

[33] D. Mudzingwa and R. Agrawal. A study of methodologies used in intrusion detection and prevention systems (idps). In *2012 Proceedings of IEEE Southeastcon*, pages 1–6.

[34] Intrusion Detection Working Group. `http://gost.isi.edu/cidf/`.

[35] S. Y. Lim and A. Jones. Network anomaly detection system: The state of art of network behaviour analysis. In *2008 International Conference on Convergence and Hybrid Information Technology*, pages 459–465.

[36] P. Garca-Teodoro, J. Daz-Verdejo, G. Maci-Fernndez, and E. Vzquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(12):18–28, 2009.

[37] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51:3448–3470, 2007.

[38] A. Kind, M. P. Stoecklin, and X. Dimitropoulos. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110–121, 2009.

[39] K. Limthong, F. Kensuke, and P. Watanapongse. Wavelet-based unwanted traffic time series analysis. In *2008 International Conference on Computer and Electrical Engineering*, pages 445–449.

[40] Aaron Beach, Matthew Modaff, and Yan Chen. Network traffic anomaly detection and characterization.

[41] Ding Meimei and H. Tian. Pca-based network traffic anomaly detection. *Tsinghua Science and Technology*, 21(5):500–509, 2016.

[42] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 14–23.

[43] Leonid Portnoy, Sal Stolfo, and Eleazar Eskin. Intrusion detection with unlabeled data using clustering. volume In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001, pages 5–8.

[44] Gerhard Munz, Sa Li, and Georg Carle. Traffic anomaly detection using k-means clustering.

[45] Wu Gaoxiang, Wei Songjie, Luo Na, and Yang Ling. Capturing and characterizing network actions of mobile applications for behavior consistency. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, pages 898–905.

[46] Taeeun Kim, MyoungSun Noh, Kyungho Chung, and Chaetae Im. *A Study on Context Information Collection for Personal Mobile Device Identification in BYOD and Smart Work Environment*, pages 104–109. Springer International Publishing, Cham, 2015.

[47] C. Zhang, Y. Hu, X. Zhu, Z. Guo, and J. Huang. Anomaly detection for user behavior in wireless network based on cross entropy. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 1258–1263.

[48] Steven Hofmeyr and Stephanie Forrest. Immunity by design: An artificial immune system. 1999.

[49] Y. Zhang, C. Liu, and H. Qin. Artificial immunity-based anomaly detection of network user behavior. In *2013 Ninth International Conference on Natural Computation (ICNC)*, pages 644–648.

[50] Wei Hao, X. Chen, and Wang Chao. User behavior analyses based on network data stream scenario. In *2012 IEEE 14th International Conference on Communication Technology*, pages 1017–1021.

[51] Dongwan Kang, Taeeun Kim, Jooyoung Kim, and Hwankuk Kim. A study on anomaly behavior analysis using bayesian inference in byod environment. In *Security Industry Technology Division KISA(Korea Internet&Security Agency)*.

[52] R. Vaarandi. Detecting anomalous network traffic in organizational private networks. In *2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, pages 285–292.

[53] Vern Paxson. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[54] FuzzyWuzzy: Fuzzy String Matching in Python. `https://github.com/seatgeek/fuzzywuzzy`.

[55] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[56] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017.

[57] A W. Moore and D Zuev. Discriminators for use in flow-based classification. 01 2005.

[58] Y. Kumano, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka. Towards real-time processing for application identification of encrypted traffic. In *2014 International Conference on Computing, Networking and Communications (ICNC)*, pages 136–140, Feb 2014.

[59] S. Sudman and N.M. Bradburn. *Asking Questions: A Practical Guide to Questionnaire Design*. Jossey-Bass social and behavioral science series. Jossey-Bass, 1986.

[60] Elizabeth Fanning. Formatting a paper-based survey questionnaire: Best practices. 10, 01 2005.

[61] N. Thayer-Hart, J. Dykema, K. Elver, N.C. Schaeffer, J. Stevenson, and Office of Quality Improvement. University of Wisconsin. *Survey Fundamentals: A Guide to Designing and Implementing Surveys*. University of Wisconsin, 2010.

[62] Douglas Steinley and Michael J. Brusco. Selection of variables in cluster analysis: An empirical comparison of eight procedures. *Psychometrika*, 73(1):125, Aug 2007.

[63] Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, Jun 1985.

[64] Determining the number of clusters in a data set. `https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set`.

[65] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(Supplement C):53 – 65, 1987.

[66] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. 63:411–423, 02 2001.