Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science

DIPLOMA THESIS AGREEMENT

Student: Kubeša David

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: Identification of network users by profiling their behavior

Guidelines:

1. Review the state-of-the art methods for traffic profiling and host profiling with special attention to how to represent the behavior of computers.
2. Propose and implement a method to create and process host profiles over time.
3. Propose a method to identify the profile of the same user in other traffic.
4. Experimentally evaluate if the proposed solution on both equally distributed and real capture datasets.
5. Critically analyze the results and propose further extensions of the solution with respect to its applicability to the StratosphereIPS project.

Bibliography/Sources:

[1] Kim, Hyunchul, et al. "Internet traffic classification demystified: myths, caveats, and the best practices." Proceedings of the 2008 ACM CoNEXT conference. ACM, 2008.
[2] Sommer, Robin, and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection." Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, 2010.
[3] Karagiannis, Thomas, et al. "Profiling the end host." International Conference on Passive and Active Network Measurement. Springer Berlin Heidelberg, 2007.
[4] Minarik, Pavel, Jan Vykopal, and Vojtech Krmicek. "Improving host profiling with bidirectional flows." Computational Science and Engineering, 2009. CSE'09. International Conference on. Vol. 3. IEEE, 2009.
[5] Karagiannis, Thomas, Konstantina Papagiannaki, and Michalis Faloutsos. "BLINC: multilevel traffic classification in the dark." ACM.

Diploma Thesis Supervisor: Ing. Sebastián García

Valid until the end of the winter semester of academic year 2018/2019

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

prof. Dr. Michal Pěchouček, MSc.
Head of Department

prof. Ing. Pavel Ripka,CSc.
Dean

Prague, June 21, 2017

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Master's thesis

Identification of network users by profiling their behavior

Bc. David Kubeša

Supervisor: Ing. Sebastián García, Ph.D.

Study Programme: Open Informatics

Field of Study: Artificial Intelligence

May 24, 2018

Aknowledgements

I would like to thank my supervisor Ing. Sebastián García, Ph.D. for the patience and for the help with writing this thesis. I would also like to thank my parents and my girlfriend Zuzana Vozárová for support during my study.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on May 24, 2018 ....

Abstract

The precise identification of users in the network at different moments in time is a well known and difficult problem. Identifying users by their actions (and not their IP addresses) allows administrators to apply policy controls on users, to find intruders that are impersonating legitimate users, and to find anomalous user behaviors that could be due to malware infections. More importantly, the behavioral analysis of users actions raises important moral questions about the power to identify users in unknown networks. This thesis explores this question by trying to identify users by converting the user's behavior into user's profiles. These profiles are time-dependent and they have dozen of features. By using the traffic of known past users in our dataset, it was possible to create and store their behavioral profiles. The profiles were created by extracting features from NetFlow data, and therefore no payload was used. The decision to only use NetFlows made this research much more challenging since there were less data. After studying the behaviors, we designed a comparison model that it is a similarity metric between users profiles. The profiles are compared one-to-one and also in sequential groups. The comparison of groups of profiles is the base for the user to user classifier. These methods were verified on experiments that used one of the largest labeled datasets currently available in the area, consisting in more than one month of real traffic from 19 known and verified normal users. All our tools were published online, including the tools to visualize and compare users. Results show that we can identify our users with 60% of accuracy and 90% precision. The success of this method mostly depends on how well we can compare two user profiles. A small improvement can lead to improvement in user detection.

Abstrakt

Přesná identifikace uživatelů na síti v různém časových úsecích se řadí mezi známé těžké problémy. Identifikace uživatelů na základě jejich akcí a ne jejich IP adress umožnuje administrátorům sítí aplikovat sofistikovanější pravidla pro kontrolu uživatelů. Tento přístup také umožnuje identifikaci nežádoucích uživatelů, co se vydávají za jiné, např. kvůli nákaze malwarem. Analýza uživatelských akcí vyvolává morální otázky, protože umožnuje identifikaci uživatelů v rozdílných sítích. Tato práce se snaží zachytit vzorce chování uživatele do uživatelských profilů. Tyto profily jsou závislé na čase a jsou vytvořené z velkého množství příznaků. Za použití záznamu sítě od známých uživatelů v našem datasetu bylo možné tyto profily vytvořit a uložit. Tyto profily byly vytvořené na základě NetFlows ze sítě. Obsah packetů proto není pro profilování použit. Používání pouze NetFlows dělá tento výzkum těžší. Porovnáváme nejdříve dvojce profilů a po té až uživatele jelikož hlavní porovnávací model pro porovnání uživatelů je založen na provnání těchto uživatelských profilů. Naše experimenty probíhali na datasetu 19 prověřených uživatelů, což je jeden z největších datasetů tohodle typu. Naše výsledky ukázaly, že jsme schopni identifikovat uživatele s 60% správností a s 90% přesností. Jak moc dobře naše metoda funguje záleží hlavně na tom, jak moc dobře dokážeme porovnat dva uživatelské profily. Malé zlepšení v tomto porovnání vede následně k lepší identifikaci uživatelů.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The precise identification of users in a network is the well known and widely research topic. The motivations range from protecting the same user while changing WiFi hotspots, to apply security restrictions to users and not IP addresses inside organizations, to find the anomalous behaviors of users that were impersonated by attackers that took control through malware infections. The correct identification of users is of huge importance for the functionality and security of computer networks.

The research community has been approaching this problem from different perspectives and for different purposes. The most effective works seem to use graphs for capturing users behavior [1]. The main idea of using graphs is that graphs capture flow information and inter-flow dependencies of the user's actions. Graphs allow a compact representation of the user behavior. The main problem with graphs seems to be the reduced amount of information that they can handle for the relationships.

Apart from graphs, the community has explored the analysis of behaviors by analyzing the HTTP headers sent by the users [2] and other approaches dealing with packet content. Some cases involve the analysis of bidirectional flows [3], but the resulting features were analyzed manually.

The analysis of behaviors also has a commercial interest due to its probably prediction capabilities. Some of the commercial approaches explored the problem trying to model user traffic from the NetFlows. One of the systems developed by Cisco [4], called FireSIGHT, is supposed to analyze network features, such as the number of connections, to raise alerts when an anomaly is detected. However, these detections are still based on static rules.

The most significant drawback of these approaches seems to be that the behavior of the users is defined by manually-crated rules. A fundamental difference of our thesis is that previous research seems to focus on creating *groups of users*, while we focus on profiling *each individual user*. As far as we know, this thesis is one of the first works to model individual users by means of their behavioral characteristics with the purpose of detecting them.

The working hypothesis of this thesis is that it is possible to find the same user on a different network by profiling and modeling their behavior in the network. To profile the behavior of users, this thesis starts by obtaining network flows [5] generated by the user, then separating the flows in hourly ranges, followed by extracting specific features from the flows and finally creating a unique behavioral profile for each hour. Since the flows do not contain any payload, we do not use any content for the creation of the profiles. The use of flows for profiling users helps us work in a more realistic scenario since most organizations are unwilling to share the content of their networks. However, the lack of precise information about what is being transferred poses a challenge to our method. We propose to identify users in the network by computing a distance measure between profiles of users. The distance measure is composed of two parts: first, a distance between two profiles, where we compared each feature. Second, a more generic distance between two *periods* of time, that we called a comparison of *users*. This more generic distance uses the profile-to-profile distance for each pair of profiles, but then it aggregates all the comparisons in the period and obtains a final decision regarding all the profiles. One of the most important learnings of our method was that the comparison of profiles *must* is done between profiles of the *same hour* on *different days*. Comparing profiles that are adjacent in time would not regard the same results. This selection of comparisons is one of the most important contributions of this thesis: the actions of a user one day are **most** similar to the actions done another day, **but during the same hour**.

In this work, we use network flows generated by the Argus tool [6]. A flow, in terms of the NetFlow standard, is usually defined as the group of uni-directional packets that share a common key. The key is typically the source IP address, the source port, the destination IP address, the destination port, and the protocol. In the NetFlow standard, all flows are uni-directional, which means that a usual communication is split in two flows. Also, NetFlows do not store payloads. The Argus tool generates a variant of flows, which are bi-directional. Bi-directional flows group together all the packets in both directions, i.e. from the client to the server and from the server to the client. We use Argus to generate bidirectional flows because of its performance and proper management of the packets. From now on each bi-directional flow will be referenced as *binetflow*.

A traffic capture is a continuous group of packets on the network. The *binetflows* capture is also a continuous group of flows, that was one flow on each line. However, for our work it is more convenient to split the binetflow capture into time windows so they can be analyzed individually. In this work, we create profiles that last one hour. This time-window is long enough to capture unique user behavior and short enough to capture changes in behavior during the day. On each one hour time-window, we extract 42 features. The design of the features is done to capture specific characteristics of the behavior. The main design characteristic of the features is made over the ports used in the traffic, and on the contacted IP addresses by the host being profiled. These features are probability distributions.

All the profiles consists of the same amount of features. When two profiles are compared, we need to compare all selected features to each other. Every comparison contributes to know how similar both profiles are. Every comparison between profiles obtains the same amount of results as the number of features because we obtain a similarity coefficient per feature. From the comparisons of each features, we need to decide if those two profiles are similar or not.

We used similarity coefficients of two probabilistic distributions for comparing profiles per feature. For the purpose of detecting similarity of two profiles, the similarity coefficients are taken into account as features for the classifier. The goal of the classifier is to find if two profiles are similar or not. The comparison of similar profiles is defined as the comparison between two profiles from the same user, from the same hour, from different days. The similar profiles in the profiles comparison are defined as a comparison between two profiles from the same user from the same hour from different days. The different profiles are defined as a comparison between two profiles from the different user from different days and the same days.

When we train the profile to profile classifier, we can tell if two profiles are similar or not. However, usually there is more data than one hour for each profile. The dataset used in this thesis consist of one and half month of traffic of users in our network. Our goal is to match new users to the known users. This user identification can be useful when the network administrator wants to be able to detect intruders impersonating some of the users. The profiles also can serve as the platform for the anomaly detection. As the rapid change in the profile can mean some sort of intrusion.

Based on profile to profile classifier, we build the user to user classifier. Every user consists of several profiles. We use the information about each user from its multiple profiles combined. We have new users that we want to match to all users we know from the dataset of the profile to profile classifier.

We compare profiles from these new users to all users we know from the dataset of the profile to profile classifier. There are many profiles comparisons that need to be taken into consideration for matching the users. We average all comparisons and take a decision based on this value. The user with the minimal value, here with the highest probability, is selected as same. To avoid false defections we decided to create a threshold rejecting matching users with too high minimum value.

Our dataset consists on the capture of 19 IP addresses during a month and half. The capture was done in one of the networks of the Czech Technical University in Prague. For privacy reasons all our dataset consist only of flows, and not packets. The span of the capture goes from the 1st of September 2016 to the 15th of October 2016.

Since we present two different algorithms (profile to profile, and user to user), there are two sets of experiments. The first set of experiments measures how well the profile to profile classifier works. The second set of experiments, measures how well the user to user classifier works. These experiments help to answer the question of if these methods can help identify a new unknown user as one of the users known to the algorithm.

Results show that the profile to profile classifier is capable of deciding, with an accuracy of 86%, if two profiles are similar or not, in other words if they come from the same user or not. The results from the user to user comparison show that if the user has sufficient amount of data, the algorithm can identify if they are the same user with 60% accuracy.

The results seem promising in that we can identify users only from NetFlows without any knowledge about the payload. The contributions of this thesis are

- We presented a way how to create time-based profiles from the binetflows capture.
- The time-based profiles allowed us to analyze larger captures thanks to the visualization of the profiles.
- According to the observation of the similarity between the profiles we developed a per features similarity coefficients.
- We designed the profile to profile classifiers based on these similarity coefficients.

- We showed that our method of detecting users in our dataset is quite reliable.

The rest of the thesis is divided as follows: Chapter 2 describes in detail how our problem and similar problems are solved by others. Chapter 3 describes every step needed for the profile to profile classifier together with the classifier results.Chapter 4 is about using many profile comparisons to get more reliable results than comparison only between two profiles. These many profile comparisons allow us to detect same users more reliably.

Chapter 2

Related work

The problems of using machine learning in the network security domain are complex and difficult to address [7]. Despite that some analysis agree that the real root cause of security problems is more economic that technical [8], there are plenty of technological improvements that can heavily impact the overall security of our networks and users. In particular, we believe that the more related with statistical features a solution is, the quicker it becomes irrelevant and useless. This is why this thesis is focusing on the behavior of users as a high level perspective and not in the contents of the networks, or the domains accessed.

In order to profile and identify the behavior of users, the usage of the port numbers for the network connection has been always a good option [7]. Given the good results and the descriptive nature of port numbers, these features were incorporated in this thesis. Port numbers have shown to be good for identifying the services accessed by the user, to identify the application making the request, and for identifying the operating system used by the computers.

One of the main problems of the domain is that each detection should be explained to have a complete understanding of the problem. This is usually a hard task that needs experts on the field. This explanation is also dependent on the type of classification methods used. Some of the most common internet traffic classification methods were shown to work quite well for this [9]. The comparison of ports-based, host-behavior-based, and flow-features-based is presented. The comparison was made on an extensive dataset to strengthen the comparison.

The classification of the footprints leaved by the transport layer was suggested in the BLINC paper [10]. The behavior of the source host in analyzed with different techniques for the purpose of detection. The analysis was divided into the three levels - Social, Function, and Application level. The social behavior describes whom the host is communicating to, and also the popularity of the host among others. The functional level is for identifying if this host is consuming or providing service. The transport layer level presented separates the traffic in 4-tuples (source and destination IP addresses, and source and destination ports). The behavior of the transport is identified using pre-prepared templates of the traffic called graphlets based on the 4-tuples.

The original ideas presented in BLINC were also studied and improved by others [1] base on their work on graphlets. Graphlets do not need to be predefined, they are the primary data used to describe the host. The main idea is that graphs can capture flow information and inter-flow dependencies. Graphlets are capable of updating themselves in time in an unsupervised way to achieve better profiling. Graphlets allow compact representation of the user behavior.

Other papers worked in improving the profiling of host profiling with bidirectional flows [3]. For this, a custom type of flows was defined. The paper uses a pairing algorithm to separate requests, replies and single flows without replies. Flow pairs are connected, allowing the method to have a very similar data to the data provided by Argus, the tool used in this thesis. This paper uses very similar features to those used in our work, describing the hosts for a later identification. However, this paper do not tries to compute a distance between features, and it only tries to identify clients and servers on the network by creating server and client profiles.

The profiling of users and hosts has many variations and approaches. A common selection of data are the HTTP requests [2]. The approach used by this paper is very similar to our own, but only focusing on the type of features that can be extracted from the HTTP protocol (HTTP logs, here using Cisco logs). The main deciding feature in this work are domains visited by IP. For the distance between domains visited by two users, the paper used the technique called tf-idf (term frequency–inverse document frequency) combined with an adjusted cosine distance. In this work the information about visited domains cannot be used as this work is based on binetflows that do not have these data. Among the differences with our work is the amount of features used, together with the use of the Bhattacharyya coefficient instead of tf-idf. Our selection of features and techniques proved better in the results. The dataset in his work is from a corporate network, while the data-set used in this work is from a university network.

One of the commercial use of the traffic profiles is the Cisco's FireSIGHT [11]. It is based on analyzing features for example number of connections and raise the alert if any anomaly behavior is detected based on correlation rules specified. This approach requires manual setup and usually one week of data for learning.

Chapter 3

Classification of individual profiles

This Chapter analyzes the *binetflows* dataset and the selection of the features that are used later in the time-based profiles of users. The visualization of every feature in the profiles is also shown in this Chapter. The visualization is useful for getting a better idea of how well the features model the behavior of the users during the day. From the visualization, we observe the similarities between the features in the profiles. This observation allows us to use a similarity coefficient on every feature in the profile. Our profile to profile classifier is later trained on these coefficients to detect if two profiles are similar or not. The output of the profile to profile classifier is the probability of two compared profiles being similar or not.

3.1 Binetflows dataset

The base network unit for this thesis is network flows. A flow, in terms of the NetFlow standard, is commonly defined as the group of uni-directional packets that share a common key. The key is typically the source IP address, the source port, the destination IP address, the destination port, and the protocol. In the NetFlow standard, all flows are uni-directional, which means that usual communication is split into two flows. The NetFlows do not store its payload. The flows used in this thesis are generated by the Argus tool [6]. Argus generates a different type of flow that is bi-directional. Bi-directional flows group together all the packets in both directions, i.e., from the client to the server and from the server to the client. Argus was chosen to generate bidirectional flows because of its performance and proper management of the packets. These bi-directional flows are referred in this thesis as *binetflows*.

The example flows parsed from the csv format to the table are in Table 3.1

StartTime	Dur	Proto	SrcAddr	Sport	Dir	DstAddr	Dport	State	sTos	dTos	TotPkts	TotBytes	SrcBytes	SrcPkts	Label
2016/10/22 00:01:39.590784	0.000000	tcp	110.35.219.245	25767	->	147.32.83.146	23	S_	0		1	64	64	1	
2016/10/05 15:37:16.620855	0.318896	tcp	147.32.83.102	54744	->	134.170.104.154	80	SPA_SPA	0	0	8	2660	2290	5	

Table 3.1: Binetflow example

Our dataset consists of 7,010,449 flows. We have a list of 19 IPs of verified users.[1]

[1]Thanks to the work of our coworker František Strašák in the Stratosphere Project https://

The dataset consists of the capture from 1st of September 2016 to 15th of October 2016. There are some missing days in the dataset. Therefore there are 34 days and not 45. The dataset should be long enough to capture user behavior patterns. We mainly analyze the hour and daily behavior patterns.

3.1.1 Generation of the Binetflows dataset

The dataset used in this work comes from the Czech Technical University in Prague (later referenced as CTU). The flows were captured in the main output routers of the University and were generated by Argus. The dataset used in this thesis is a subset of all the traffic obtained in the University. The dataset consists of 19 different IP addresses, used by 19 different users during 34 days. The exact configuration of Argus used to capture the dataset can be found in the GitHub repository [12].

3.1.2 Detection of established and not established flows

This section is about detection of established and non-established flows for TCP and UDP in a *binetflow* capture.

The TCP headers in the flows are represented by the following state letters:

- S means Synchronization
- A means Acknowledged
- R means Reset
- P means Push
- F means Fin
- U means Urgent

The state column sometimes also contains word CONN or EST. This means that the connection was established. REQ means that the connection was not established as the only request for connection was sent. The CONN, EST, and REQ are available if the Argus is sure what connection type it is. Otherwise, the state letters are provided. (For example, SPA_SPA from Table 3.1 means that client sent Syn, Push and Acknowledged packets(in any order) and the server responded with same packets type.)

For the detection of the connection type these conditions are checked in corresponding order:

1. Detecting established connection:
    - State column contains CONN or EST
    - State column of connection information of source IP address contains S, and the destination IP address contains S and A

`stratosphereips.org/` who matched the IP addresses to the real users on the faculty network.

2. Detecting not-established connection:
    - State column contains REQ
    - State column of connection information of source IP address contains S, and there is no answer.
    - State column of connection information of source IP address contains S, and the response is RA - Reset/Acknowledged.

The UDP protocol is stateless so is more challenging to identify established connection. We can only try to recognize if the UDP flow contains the first datagram or not. For the UDP, the detection of established and not established connection is following:

- The flow is recognized as an established connection if the state column contain CONN
- The flow is recognized as a not-established connection if the state column contains INT

3.2 Profiles from *binetflows*

To be able to analyze user behaviors more easily, we decided to create time-based profiles. These profiles are made from continuous *binetflows* captures. In this work, the time interval of the profile is one hour. The one-hour interval was chosen empirically as the interval that is sufficiently long to capture individual user behavior, but not too long to lose the changes of the behavior of the user during the day. Every hour we measure the selected features described. The features in the profiles are separated into the following groups:

- Summary features in subsection 3.2.1.
- Location -based features in subsection 3.2.2.
- Connection-based features in subsection 3.2.3.
- Port-based features in subsection 3.2.4.
- IP addresses-based features 3.2.5.
- Features based on not answered connections 3.2.6.

3.2.1 Summary features

The purpose of these features is to have a quick idea what the user was doing in the selected hour. These features are:

1. clientNumberOfDistinctOrganizationsEstablished This feature holds the total number of distinct organizations contacted where the connection was from the tracked IP and the connection was established.

2. clientNumberOfIPFlows This feature summarizes all the flows from the tracked IP.

3. serverTotalNumberOfTransferedData This feature summarizes a total number of transferred data to the tracked IP.

4. clientNumberOfConnectionsNotEstablished This feature summarizes a number of connections that were from the tracked IP and were not established.

5. serverNumberOfIPFlows This feature summarizes all the flows to the tracked IP.

6. numberOfIPFlows This feature summarizes all the flows of the IP.

7. clientNumberOfDistinctCountriesEstablished This feature summarizes a number of distinct countries contacted by tracked IP where the connection was established.

8. serverNumberOfConnectionsNotEstablished This feature summarizes a number of connections that were not established.

9. clientTotalNumberOfTransferedData This feature summarizes a total number of the transferred data from the tracked IP.

3.2.2 Location-based features

The purpose of these features is to see to which countries is user connecting and from which countries connections are made to the user. The data of these features are based on the Whois data.

1. clientDictNumberOfDistinctCountriesEstablished This feature tracks connections of the selected IP to the countries where the connection was established.

2. serverDictNumberOfDistinctCountriesEstablished This feature tracks connections to the selected IP from the countries where the connection was established.

3.2.3 Connection-based features

The purpose of the connection features is to distinguish users based on to where are they connecting. The port of the connection does not matter. The tracked port features are:

1. clientDictOfConnectionsTCPEstablished This feature holds the dictionary of connections where the connection was from the selected IP, the protocol used was TCP, and the connection was established.

2. clientDictOfConnectionsTCPNotEstablished This feature holds the dictionary of connections where the connection was from the selected IP, the protocol used was TCP, and the connection was not established.

3. clientDictOfConnectionsUDPEstablished This feature holds the dictionary of connections where the connection was from the selected IP, the protocol used was UDP, and the connection was established.

4. clientDictOfConnectionsUDPNotEstablished This feature holds the dictionary of connections where the connection was from the selected IP, the protocol used was UDP, and the connection was not established.

5. clientDictClassBnetworksTCPEstablished This feature is a dictionary of contacted class B networks where the connection was from the selected IP, the protocol used was TCP, and the connection was established.

6. clientDictClassBnetworksTCPNotEstablished This feature is a dictionary of contacted class B networks where the connection was from the selected IP, the protocol used was TCP, and the connection was not established.

7. clientDictClassBnetworksUDPEstablished This feature is a dictionary of contacted class B networks where the connection was from the selected IP, the protocol used was UDP, and the connection was established.

8. clientDictClassBnetworksUDPNotEstablished This feature is a dictionary of contacted class B networks where the connection was from the selected IP, the protocol used was UDP, and the connection was not established.

9. serverDictOfConnectionsTCPEstablished This feature holds the dictionary of connections where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

10. serverDictOfConnectionsTCPNotEstablished This feature holds the dictionary of connections where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

11. serverDictOfConnectionsUDPEstablished This feature holds the dictionary of connections where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

12. serverDictOfConnectionsUDPNotEstablished This feature holds the dictionary of connections where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

13. serverDictClassBnetworksTCPEstablished This feature is a dictionary of contacted class B networks where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

14. serverDictClassBnetworksTCPNotEstablished This feature is a dictionary of contacted class B networks where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

15. serverDictClassBnetworksUDPEstablished This feature is a dictionary of contacted class B networks where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

16. serverDictClassBnetworksUDPNotEstablished This feature is a dictionary of contacted class B networks where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

3.2.4 Port-based features

The goal of the port features is to see to which ports are user usually connecting to or which ports his computer typically use for opening the connection. The tracked port features are:

1. clientSourcePortTotalBytesTCPEstablished This feature holds the total number of bytes transferred through the source port where the connection was from the selected client IP, the protocol used was TCP, and the connection was established.

2. clientSourcePortTotalBytesTCPNotEstablished This feature holds the total number of bytes transferred through the source port where the connection was from the selected client IP, the protocol used was TCP, and the connection was not established.

3. clientSourcePortTotalBytesUDPEstablished This feature holds the total number of bytes transferred through the source port where the connection was from the selected client IP, the protocol used was UDP, and the connection was established.

4. clientSourcePortTotalBytesUDPNotEstablished This feature holds the total number of bytes transferred through the source port where the connection was from the selected client IP, the protocol used was UDP, and the connection was not established.

5. clientSourcePortTotalPacketsTCPEstablished This feature holds the total number of packets transferred through the source port where the connection was from the selected client IP, the protocol used was TCP, and the connection was established.

6. clientSourcePortTotalPacketsTCPNotEstablished This feature holds the total number of packets transferred through the source port where the connection was from the selected client IP, the protocol used was TCP, and the connection was not established.

7. clientSourcePortTotalPacketsUDPEstablished This feature holds the total number of packets transferred through the source port where the connection was from the selected client IP, the protocol used was UDP, and the connection was established.

8. clientSourcePortTotalPacketsUDPNotEstablished This feature holds the total number of packets transferred through the source port where the connection was from the selected client IP, the protocol used was UDP, and the connection was not established.

9. clientSourcePortNumberOfFlowsTCPEstablished This feature holds the total number of flows transferred through the source port where the connection was from the selected client IP, the protocol used was TCP, and the connection was established.

10. clientSourcePortNumberOfFlowsTCPNotEstablished This feature holds the total number of flows transferred through the source port where the connection was from the selected client IP, the protocol used was TCP, and the connection was not established.

11. clientSourcePortNumberOfFlowsUDPEstablished This feature holds the total number of flows transferred through the source port where the connection was from the selected client IP, the protocol used was UDP, and the connection was established.

12. clientSourcePortNumberOfFlowsUDPNotEstablished This feature holds the total number of flows transferred through the source port where the connection was from the selected client IP, the protocol used was UDP, and the connection was not established.

13. clientDestinationPortTotalBytesTCPEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from the selected client IP, the protocol used was TCP, and the connection was established.

14. clientDestinationPortTotalBytesTCPNotEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from the selected client IP, the protocol used was TCP, and the connection was not established.

15. clientDestinationPortTotalBytesUDPEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from the selected client IP, the protocol used was UDP, and the connection was established.

16. clientDestinationPortTotalBytesUDPNotEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from the selected client IP, the protocol used was UDP, and the connection was not established.

17. clientDestinationPortTotalPacketsTCPEstablished This feature holds the total number of packets transferred to the destination port where the connection was from the selected client IP, the protocol used was TCP, and the connection was established.

18. clientDestinationPortTotalPacketsTCPNotEstablished This feature holds the total number of packets transferred to the destination port where the connection was from the selected client IP, the protocol used was TCP, and the connection was not established.

19. clientDestinationPortTotalPacketsUDPEstablished This feature holds the total number of packets transferred to the destination port where the connection was from the selected client IP, the protocol used was UDP, and the connection was established.

20. clientDestinationPortTotalPacketsUDPNotEstablished This feature holds the total number of packets transferred to the destination port where the connection was from the selected client IP, the protocol used was UDP, and the connection was not established.

21. clientDestinationPortNumberOfFlowsTCPEstablished This feature holds the total number of flows transferred to the destination port where the connection was from the selected client IP, the protocol used was TCP, and the connection was established.

22. clientDestinationPortNumberOfFlowsTCPNotEstablished This feature holds the total number of flows transferred to the destination port where the connection was from the selected client IP, the protocol used was TCP, and the connection was not established.

23. clientDestinationPortNumberOfFlowsUDPEstablished This feature holds the total number of flows transferred to the destination port where the connection was from the selected client IP, the protocol used was UDP, and the connection was established.

24. clientDestinationPortNumberOfFlowsUDPNotEstablished This feature holds the total number of flows transferred to the destination port where the connection was from the selected client IP, the protocol used was UDP, and the connection was not established.

25. serverSourcePortTotalBytesTCPEstablished This feature holds the total number of bytes transferred through the source port where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

26. serverSourcePortTotalBytesTCPNotEstablished This feature holds the total number of bytes transferred through the source port where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

27. serverSourcePortTotalBytesUDPEstablished This feature holds the total number of bytes transferred through the source port where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

28. serverSourcePortTotalBytesUDPNotEstablished This feature holds the total number of bytes transferred through the source port where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

29. serverSourcePortTotalPacketsTCPEstablished This feature holds the total number of packets transferred through the source port where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

30. serverSourcePortTotalPacketsTCPNotEstablished This feature holds the total number of packets transferred through the source port where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

31. serverSourcePortTotalPacketsUDPEstablished This feature holds the total number of packets transferred through the source port where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

32. serverSourcePortTotalPacketsUDPNotEstablished This feature holds the total number of packets transferred through the source port where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

33. serverSourcePortNumberOfFlowsTCPEstablished This feature holds the total number of flows transferred through the source port where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

34. serverSourcePortNumberOfFlowsTCPNotEstablished This feature holds the total number of flows transferred through the source port where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

35. serverSourcePortNumberOfFlowsUDPEstablished This feature holds the total number of flows transferred through the source port where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

36. serverSourcePortNumberOfFlowsUDPNotEstablished This feature holds the total number of flows transferred through the source port where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

37. serverDestinationPortTotalBytesTCPEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

38. serverDestinationPortTotalBytesTCPNotEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

39. serverDestinationPortTotalBytesUDPEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

40. serverDestinationPortTotalBytesUDPNotEstablished This feature holds the total number of bytes transferred to the destination port where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

41. serverDestinationPortTotalPacketsTCPEstablished This feature holds the total number of packets transferred to the destination port where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

42. serverDestinationPortTotalPacketsTCPNotEstablished This feature holds the total number of packets transferred to the destination port where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

43. serverDestinationPortTotalPacketsUDPEstablished This feature holds the total number of packets transferred to the destination port where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

44. serverDestinationPortTotalPacketsUDPNotEstablished This feature holds the total number of packets transferred to the destination port where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

45. serverDestinationPortNumberOfFlowsTCPEstablished This feature holds the total number of flows transferred to the destination port where the connection was from some remote IP, the protocol used was TCP, and the connection was established.

46. serverDestinationPortNumberOfFlowsTCPNotEstablished This feature holds the total number of flows transferred to the destination port where the connection was from some remote IP, the protocol used was TCP, and the connection was not established.

47. serverDestinationPortNumberOfFlowsUDPEstablished This feature holds the total number of flows transferred to the destination port where the connection was from some remote IP, the protocol used was UDP, and the connection was established.

48. serverDestinationPortNumberOfFlowsUDPNotEstablished This feature holds the total number of flows transferred to the destination port where the connection was from some remote IP, the protocol used was UDP, and the connection was not established.

3.2.5 IP addresses-based features

The list of IP addresses where a user is attempting to connect and the connection is not established are stored in these features.

1. clientDestinationPortDictIPsTCPEstablished This feature holds contacted IP addresses contacted through the destination port where the connection was from the selected IP, the protocol used was TCP and the connection was established.

2. clientDestinationPortDictIPsUDPEstablished This feature holds contacted IP addresses contacted through the destination port where the connection was from the selected IP, the protocol used was UDP and the connection was established.

3. serverDestinationPortDictIPsTCPEstablished This feature holds contacted IP addresses contacted through the destination port where the connection was from some remote IP, the protocol used was TCP and the connection was established.

4. serverDestinationPortDictIPsUDPEstablished This feature holds contacted IP addresses contacted through the destination port where the connection was from some remote IP, the protocol used was UDP and the connection was established.

3.2.6 Features based on not answered connections

These features describes number of trials to connect to the IP addresses and number of connection attempts to connect from some distant IP addresses.

1. clientDictOfConnectionsTCPNotEstablished This feature holds the dictionary of connections where the connection was from the selected IP, the protocol used was TCP and the connection was not established.

2. clientDictOfConnectionsUDPNotEstablished This feature holds the dictionary of connections where the connection was from the selected IP, the protocol used was UDP and the connection was not established.

3. serverDictOfConnectionsTCPNotEstablished This feature holds the dictionary of connections where the connection was from some remote IP, the protocol used was TCP and the connection was not established.

4. serverDictOfConnectionsUDPNotEstablished This feature holds the dictionary of connections where the connection was from some remote IP, the protocol used was UDP and the connection was not established.

5. clientDictOfConnectionsEstablished This feature holds the dictionary of connections where the connection was from the selected IP, TCP and UDP protocols are combined and the connection was established.

6. serverDictOfConnectionsEstablished This feature holds the dictionary of connections where the connection was from some remote IP, TCP and UDP protocols are combined and the connection was established.

3.2.7 Profiles dataset

The profiles created from the *binetflows* capture are stored in the JSON file. This JSON is taken as the dataset for the creating the distances between hour profiles. The tool for creating this dataset is called parse-binet. The code together with the usage guide is located in this GitHub repository [12].

3.3 Visualization of Profiles. ManaTi Plugin

The motivation behind the profile visualization is to understand how our features characterize the user behavior. The visual analysis enables the comparison between several hour profiles and gives a better idea of what user is doing. The visualization of the profiles had an important role in selecting features that are later used in the profile to profile classifier of Section 3.6. The visualization of the profile features is part of the project ManaTi [13][2].

The goal of the ManaTi project is to develop machine learning techniques to assist an intuitive threat analyst to speed the discovery of new security problems. The ManaTi at the time of writing this thesis is capable of analyzing Weblogs and Bro [14] logs. Visualization of the profiles coming from the *bineflows* capture extends its capabilities. It adds the possibility to analyze much larger time captures at once and get a better understanding of the user behavior during the day. The visualization is implemented in the fork of the ManaTi version 0.7.1. The new tab called Profile is added to the ManaTi. The Manati fork together with the setup guide is in this GitHub repository [15]. The parse-binet[12] tool for parsing *binetflows* and producing profiles in the JSON format is part of the ManaTi fork as a sub-module.

3.3.1 Usage

Our work visualizing Profiles was included in the Profile tab of the ManaTI. The Profile tab contains all visualization related to profiles. To view the visualization user can either:

- Load profiles in form of the JSON file using the Browse button.
- Load *binetflow* capture using the Browse button.
- Open existing analysis session by selecting it in the analysis sessions tab.

When the JSON file is loaded, the Visualization of the profiles is shown. The user is able to save the profiles to the server using the SAVE! button. The name of the analysis session is by default the name of the file loaded, but can be changed by clicking on it.

When the *binetflow* capture is loaded, the user is asked to provide a file with IP addresses that should be profiled in the capture. The format of that file is described in parse-binet GitHub repository [12]. When the file is provided the *binetflow* file is processed and loaded to the server where the profiles for selected IP addresses are created. When this is done the user Visualization is shown. A user can then Download the JSON file containing the visualization using the Download button. When the analysis session containing profiles is

[2]Project Reference: 13141/830-8301351C009, AIC Group, Department of Computer Science, CTU University.

selected, the Visualization is shown. The user also can download the profiles in the JSON format using the Download button.

When the visualization itself is loaded, a user is presented with the IP addresses as a row of buttons. When the IP address is selected, the dates are shown again as the row of buttons. When the date is selected, the 24 hours of the day together with the Day summary toggle is shown. The Day summary toggle contains the general summary of an IP address during the selected day. The daily summary is visible in Figure 3.1. The purpose of the daily summary is to have a general idea how the IP behave. The graphs are based on values from the Hour summary features from subsection 3.2.1

3.3.2 Visualization of hour profiles

When the hour is selected, a user can choose from seven tabs each presenting different features. These tabs are:

- Hour Summary. The Hour summary features corresponds to the features from subsection 3.2.1 in one table. Example screen-shot in Figure 3.2.
- Regions. The regions tab contain visualization features from subsection 3.2.2 on the world map. The screen-shot is in Figure 3.3.
- Class B networks. This tab contain visualization of Class B features from subsection 3.2.3 as a donut chart. The screen-shot is in Figure 3.4.
- Port features. This tab contain visualization of features from subsection 3.2.4 as histograms. The screen-shot is in Figure 3.6
- IPs for ports. This tab contain features from subsection 3.2.5 in the table. The screen-shot is in Figure 3.6
- Not answered connections. This tab contain visualization of features from subsection 3.2.6. The screen-shot is in Figure 3.7.
- JSON. This tab contains view of the raw JSON data for further investigation. The screen-shot is in Figure 3.8.

3.3.3 Conclusion of the visualization

We presented the visual comparison that can give a better idea what user is doing during the day. In the visualization malware attacks can also be tracked. Traffic of the WannaCry malware is well visible in the Port features and Not answered connections tab for example. This tool is therefore now one of the tools used by the Civilshpere team[3] for the forensic analyst of the *binetflow* capture. Information obtained from the visualization of the hour profiles were later used in the idea of creating the distance between two-hour profiles and model the similar and not similar user behavior.

[3] https://www.stratosphereips.org/civilsphere/about/

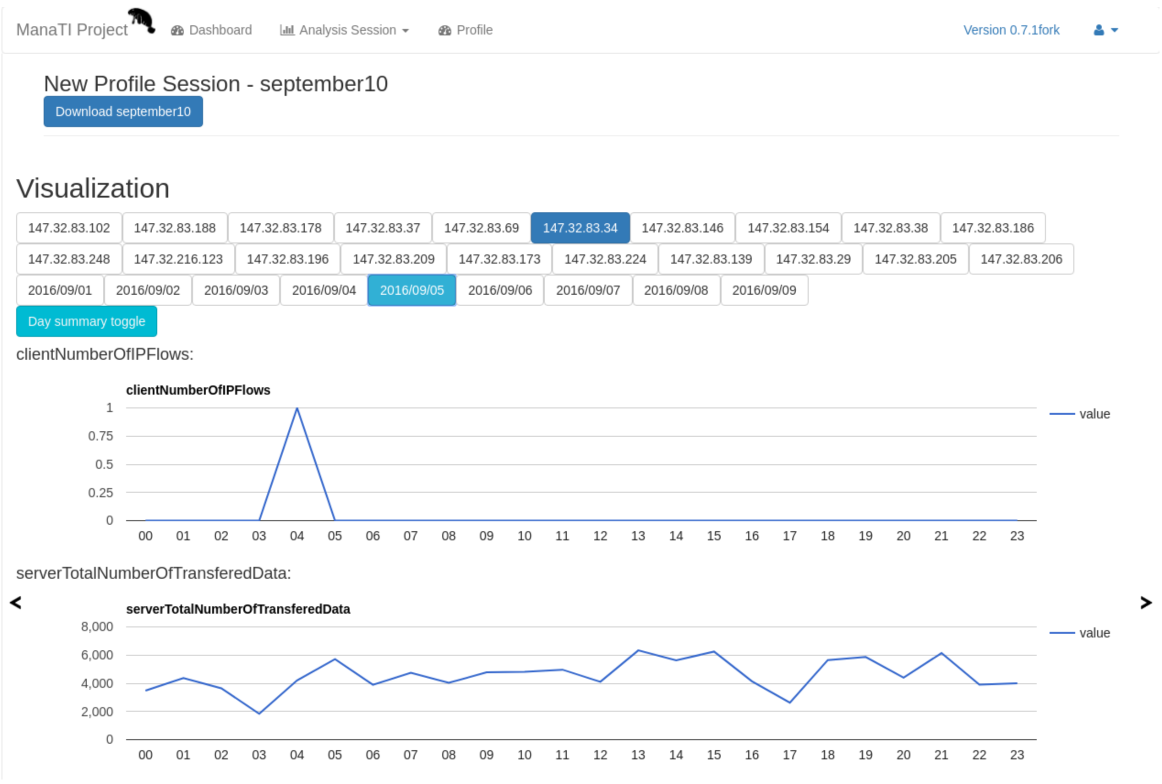
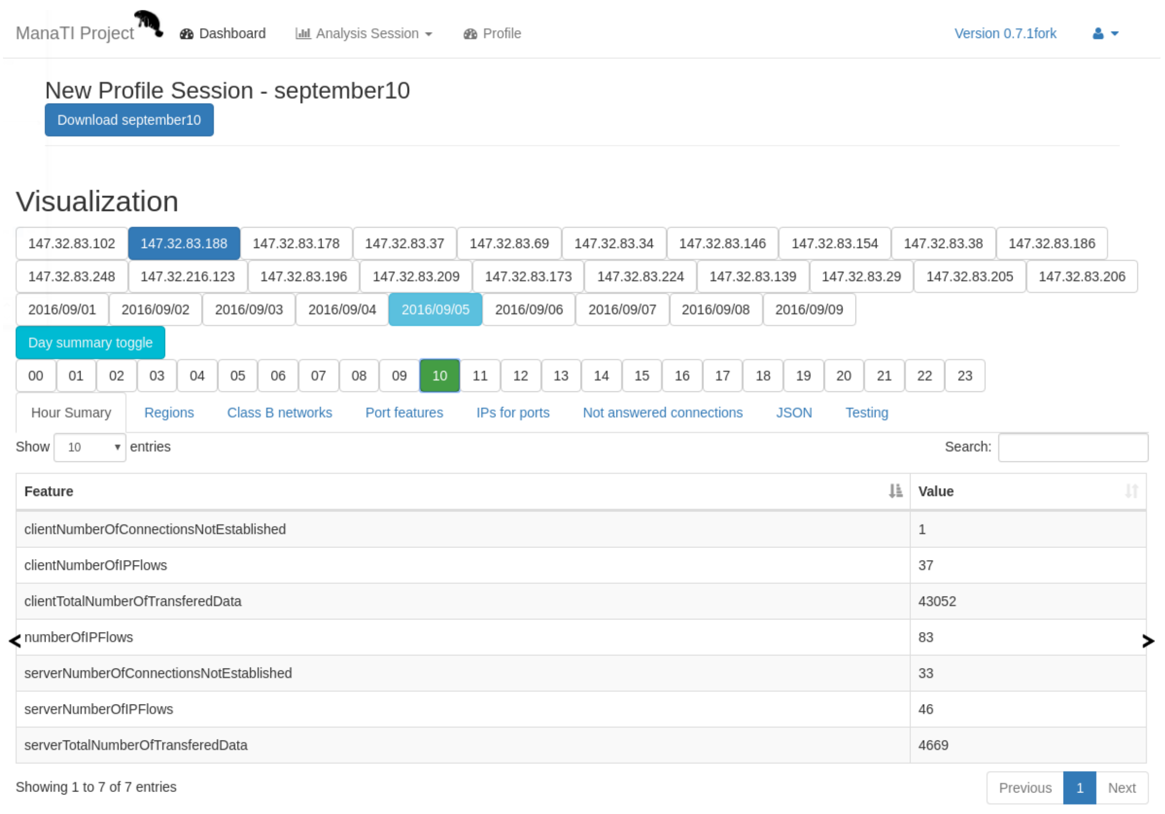


Figure 3.1: Daily summary of every hour summary feature. In here the graphs with changes are visualized. In this screenshot the "clientNumberOfIPFlows" and "serverTotalNumberOfTransferedData" are shown. "clientNumberOfIPFlows" describes the number of flows established by the client. "serverTotalNumberOfTransferedData" describes how much data were transferred to when the connection was initialized from the network. From these two features, we can judge that this IP has mainly server traffic. It can mean that the IP is a server or the user is connecting to this machine over ssh, or there is torrent running on this IP.



New Profile Session - september10

Visualization

147.32.83.102 | 147.32.83.188 | 147.32.83.178 | 147.32.83.37 | 147.32.83.69 | 147.32.83.34 | 147.32.83.146 | 147.32.83.154 | 147.32.83.38 | 147.32.83.186
147.32.83.248 | 147.32.216.123 | 147.32.83.196 | 147.32.83.209 | 147.32.83.173 | 147.32.83.224 | 147.32.83.139 | 147.32.83.29 | 147.32.83.205 | 147.32.83.206
2016/09/01 | 2016/09/02 | 2016/09/03 | 2016/09/04 | 2016/09/05 | 2016/09/06 | 2016/09/07 | 2016/09/08 | 2016/09/09
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23

Hour Sumary | Regions | Class B networks | Port features | IPs for ports | Not answered connections | JSON | Testing

Show 10 entries Search:

Feature	Value
clientNumberOfConnectionsNotEstablished	1
clientNumberOfIPFlows	37
clientTotalNumberOfTransferedData	43052
numberOfIPFlows	83
serverNumberOfConnectionsNotEstablished	33
serverNumberOfIPFlows	46
serverTotalNumberOfTransferedData	4669

Showing 1 to 7 of 7 entries

Figure 3.2: Hour summary. Here the cumulative features are shown. The complete list of these features is in subsection 3.2.1. From these summary features, we can judge that the user is active, server traffic can indicate some torrents.

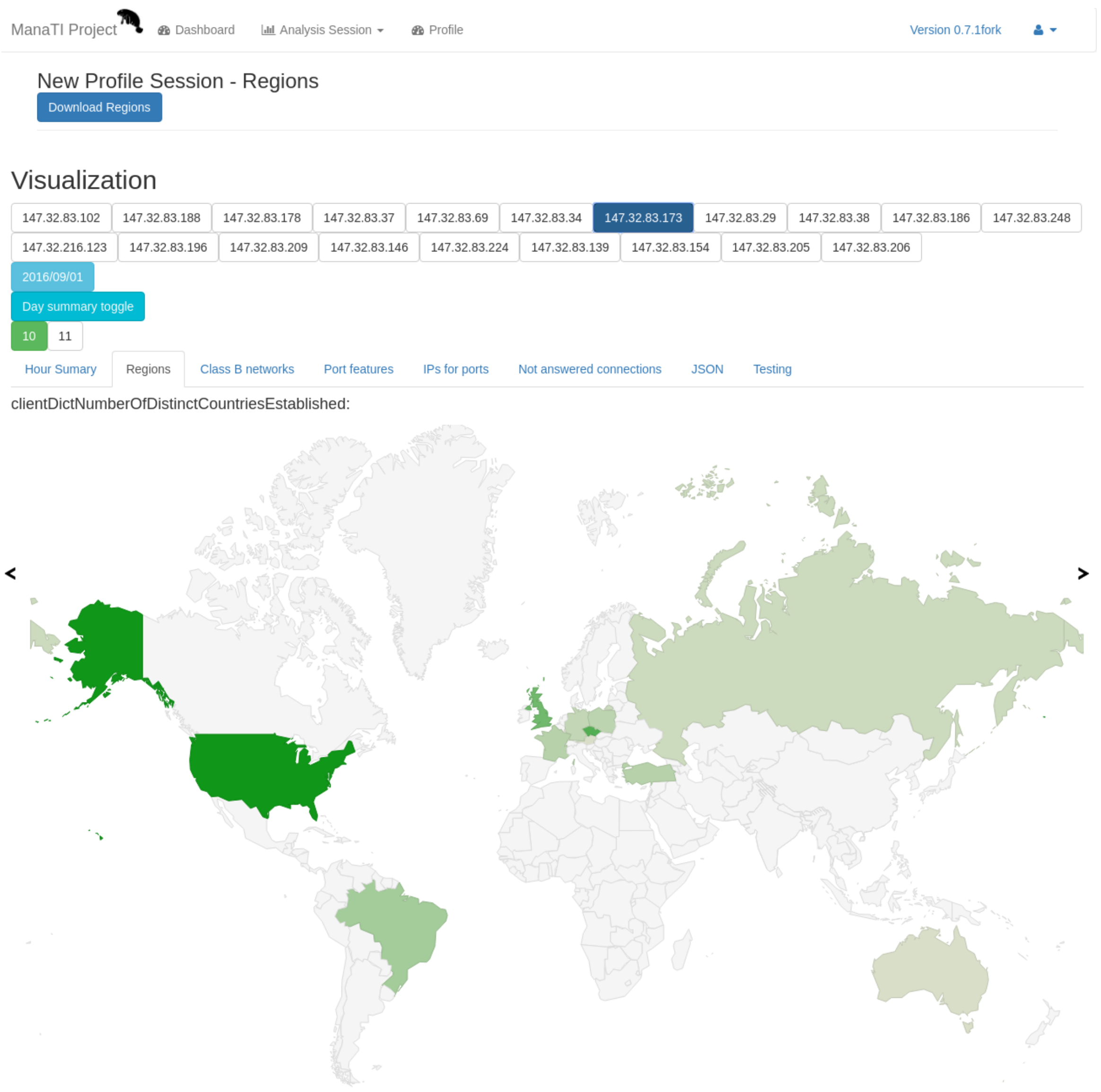


Figure 3.3: Region visualization. In here the "clientDictNumberOfDistinctCountriesEstablished" is visualized. This feature describes how many connections were made to the countries. The opposite feature - from which countries the connections were made is also implemented but not visible in this screenshot. The data of this feature are based on the Whois data. These connections look like rather normal traffic.

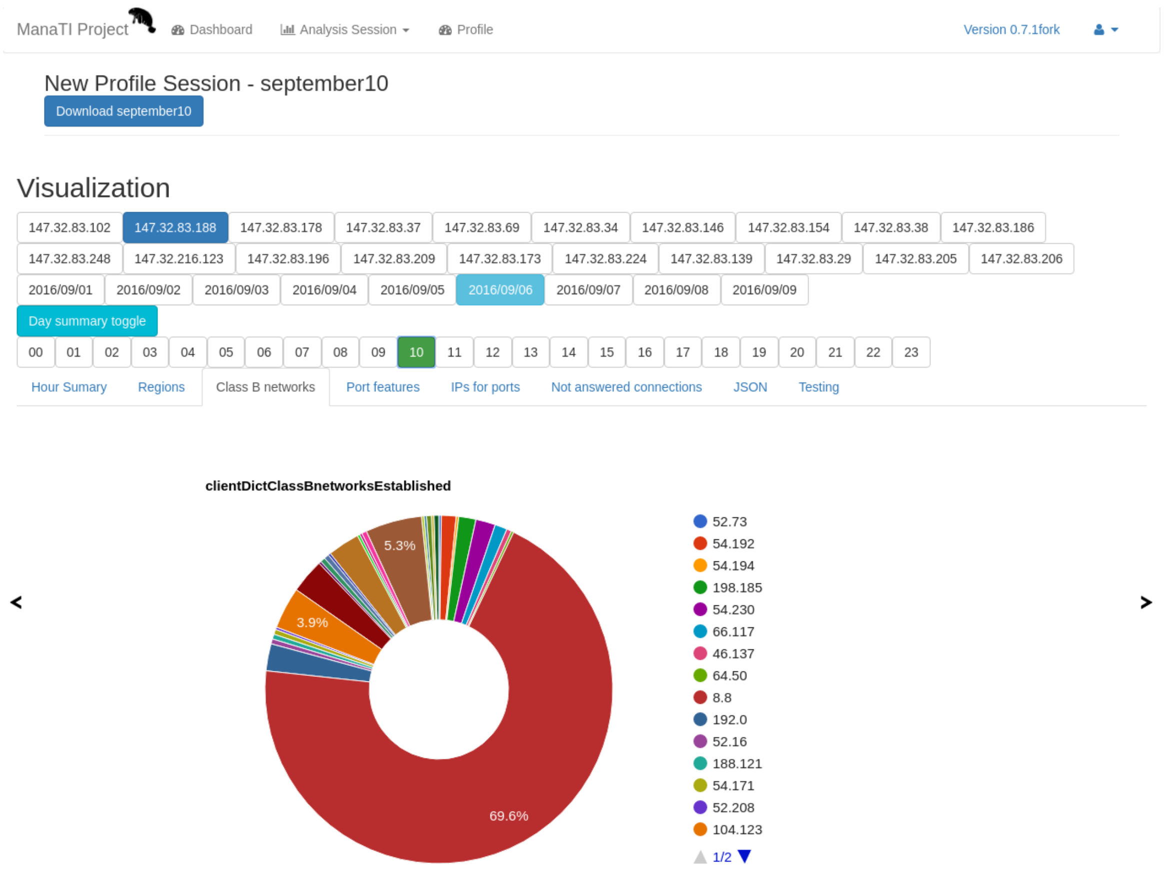
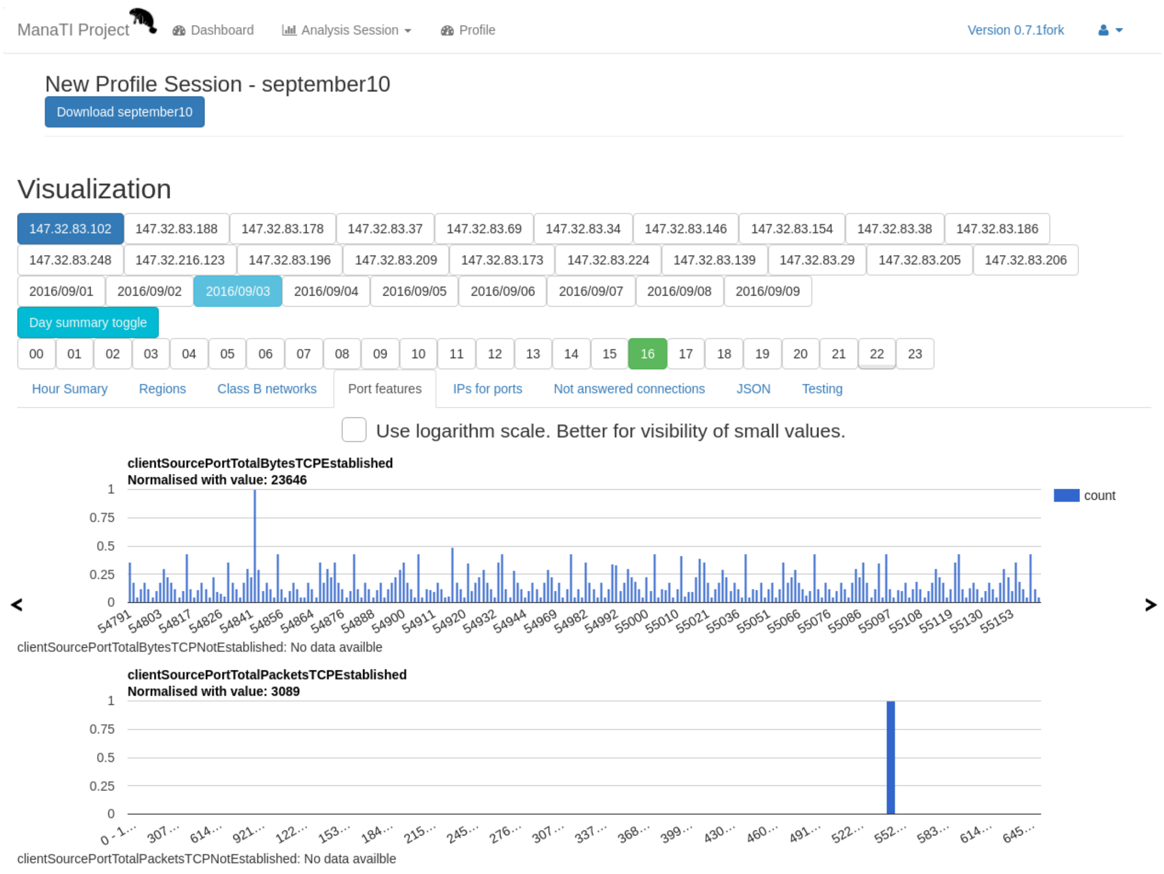


Figure 3.4: Class B networks visualization. In this case "clientDictClassBnetworksEstablished". The server feature is also visualized, but it is not visible in this screen-shot.



New Profile Session - september10

Use logarithm scale. Better for visibility of small values.

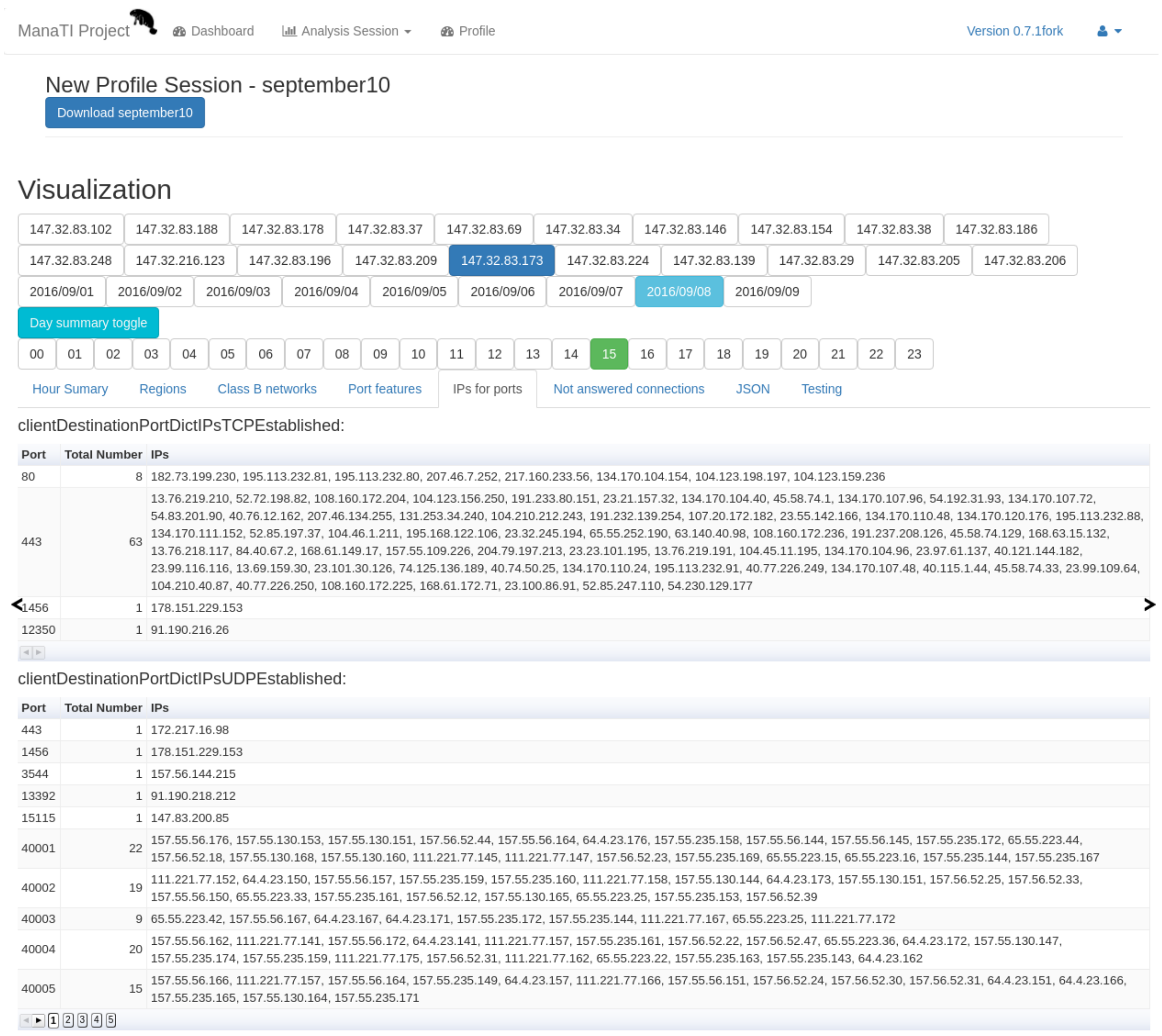
clientSourcePortTotalBytesTCPEstablished
Normalised with value: 23646

clientSourcePortTotalBytesTCPNotEstablished: No data availble

clientSourcePortTotalPacketsTCPEstablished
Normalised with value: 3089

clientSourcePortTotalPacketsTCPNotEstablished: No data availble

Figure 3.5: Port features. Two features are visible in the screen-shot. In this case features "clientSourcePortTotalBytesTCPEstablished" and "clientSourcePortTotalPacketsTCPEstablished". The second graph show histogram of ports and the first graph shows one of the bins expanded.



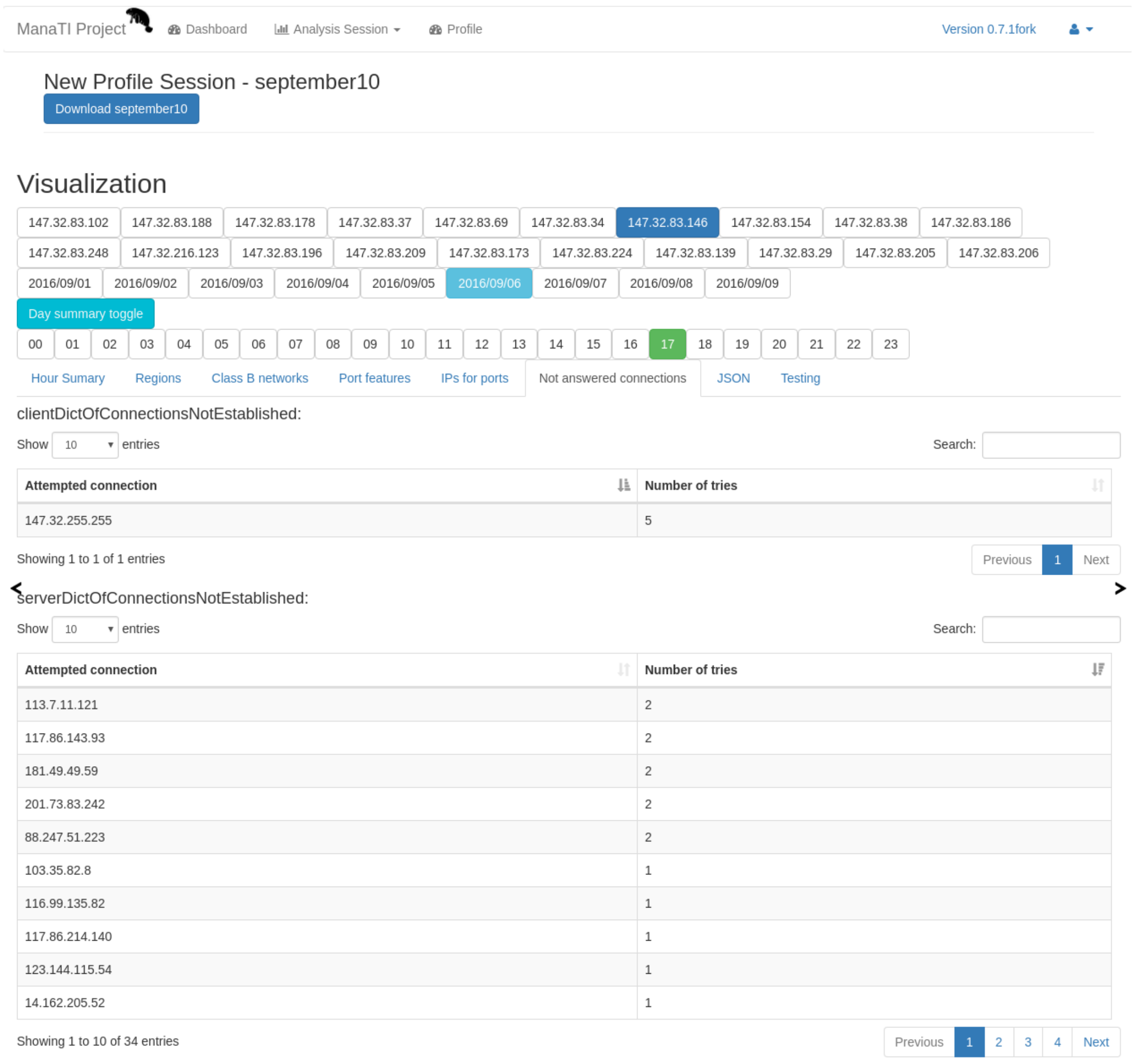
clientDestinationPortDictIPsTCPEstablished:

Port	Total Number	IPs
80	8	182.73.199.230, 195.113.232.81, 195.113.232.80, 207.46.7.252, 217.160.233.56, 134.170.104.154, 104.123.198.197, 104.123.159.236
443	63	13.76.219.210, 52.72.198.82, 108.160.172.204, 104.123.156.250, 191.233.80.151, 23.21.157.32, 134.170.104.40, 45.58.74.1, 134.170.107.96, 54.192.31.93, 134.170.107.72, 54.83.201.90, 40.76.12.162, 207.46.134.255, 131.253.34.240, 104.210.212.243, 191.232.139.254, 107.20.172.182, 23.55.142.166, 134.170.110.48, 134.170.120.176, 195.113.232.88, 134.170.111.152, 52.85.197.37, 104.46.1.211, 195.168.122.106, 23.32.245.194, 65.55.252.190, 63.140.40.98, 108.160.172.236, 191.237.208.126, 45.58.74.129, 168.63.15.132, 13.76.218.117, 84.40.67.2, 168.61.149.17, 157.55.109.226, 204.79.197.213, 23.23.101.195, 13.76.219.191, 104.45.11.195, 134.170.104.96, 23.97.61.137, 40.121.144.182, 23.99.116.116, 13.69.159.30, 23.101.30.126, 74.125.136.189, 40.74.50.25, 134.170.110.24, 195.113.232.91, 40.77.226.249, 134.170.107.48, 40.115.1.44, 45.58.74.33, 23.99.109.64, 104.210.40.87, 40.77.226.250, 108.160.172.225, 168.61.172.71, 23.100.86.91, 52.85.247.110, 54.230.129.177
1456	1	178.151.229.153
12350	1	91.190.216.26

clientDestinationPortDictIPsUDPEstablished:

Port	Total Number	IPs
443	1	172.217.16.98
1456	1	178.151.229.153
3544	1	157.56.144.215
13392	1	91.190.218.212
15115	1	147.83.200.85
40001	22	157.55.56.176, 157.55.130.153, 157.55.130.151, 157.56.52.44, 157.55.56.164, 64.4.23.176, 157.55.235.158, 157.55.56.144, 157.55.56.145, 157.55.235.172, 65.55.223.44, 157.56.52.18, 157.55.130.168, 157.55.130.160, 111.221.77.145, 111.221.77.147, 157.56.52.23, 157.55.235.169, 65.55.223.15, 65.55.223.16, 157.55.235.144, 157.55.235.167
40002	19	111.221.77.152, 64.4.23.150, 157.55.56.157, 157.55.235.159, 157.55.235.160, 111.221.77.158, 157.55.130.144, 64.4.23.173, 157.55.130.151, 157.56.52.25, 157.56.52.33, 157.55.56.150, 65.55.223.33, 157.55.235.161, 157.56.52.12, 157.55.130.165, 65.55.223.25, 157.55.235.153, 157.56.52.39
40003	9	65.55.223.42, 157.55.56.167, 64.4.23.167, 64.4.23.171, 157.55.235.172, 157.55.235.144, 111.221.77.167, 65.55.223.25, 111.221.77.172
40004	20	157.55.56.162, 111.221.77.141, 157.55.56.172, 64.4.23.141, 111.221.77.157, 157.55.235.161, 157.56.52.22, 157.56.52.47, 65.55.223.36, 64.4.23.172, 157.55.130.147, 157.55.235.174, 157.55.235.159, 111.221.77.175, 157.56.52.31, 111.221.77.162, 65.55.223.22, 157.55.235.163, 157.55.235.143, 64.4.23.162
40005	15	157.55.56.166, 111.221.77.157, 157.55.56.164, 157.55.235.149, 64.4.23.157, 111.221.77.166, 157.55.56.151, 157.56.52.24, 157.56.52.30, 157.56.52.31, 64.4.23.151, 64.4.23.166, 157.55.235.165, 157.55.130.164, 157.55.235.171

Figure 3.6: IP addresses for ports. In this case the "clientDestinationPortDictIPsTCPEstablished" and "clientDestinationPortDictIPsUDPEstablished" are visualized. These features shows which IP addresses were contacted using the selected port as a client. Some server features are implemented, but not visible in this screenshot.



New Profile Session - september10

Visualization

clientDictOfConnectionsNotEstablished:

Show 10 entries   Search:

Attempted connection	Number of tries
147.32.255.255	5

Showing 1 to 1 of 1 entries

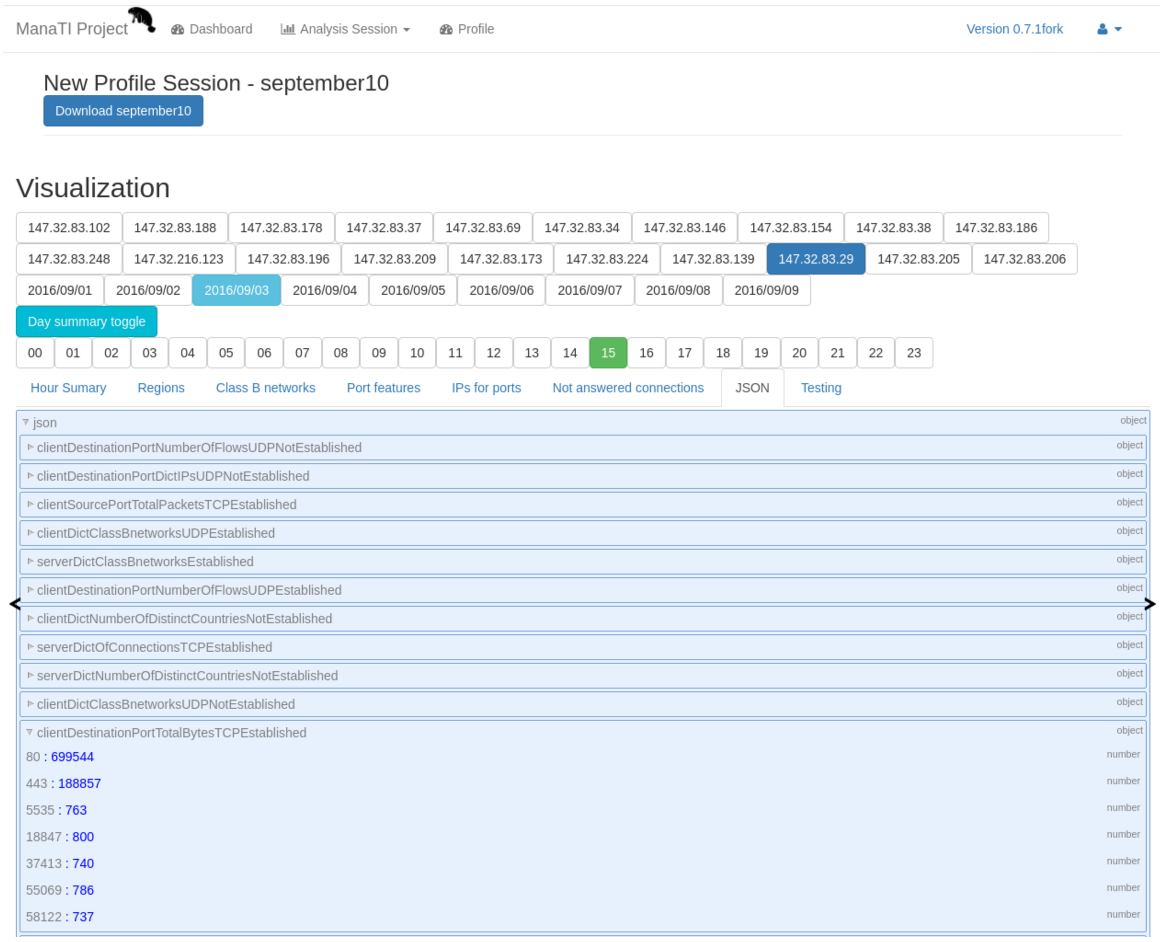
serverDictOfConnectionsNotEstablished:

Show 10 entries   Search:

Attempted connection	Number of tries
113.7.11.121	2
117.86.143.93	2
181.49.49.59	2
201.73.83.242	2
88.247.51.223	2
103.35.82.8	1
116.99.135.82	1
117.86.214.140	1
123.144.115.54	1
14.162.205.52	1

Showing 1 to 10 of 34 entries

Figure 3.7: Visualization of the not answered connections. Here the "clientDictOfConnectionsNotEstablished" and "serverDictOfConnectionsNotEstablished" features are shown. The feature "clientDictOfConnectionsNotEstablished" show which IP the user was connecting to and how many times. The "serverDictOfConnectionsNotEstablished" feature describes from which IP adresses the user was contacted with no answer.



```
json                                                      object
  clientDestinationPortNumberOfFlowsUDPNotEstablished     object
  clientDestinationPortDictIPsUDPNotEstablished           object
  clientSourcePortTotalPacketsTCPEstablished              object
  clientDictClassBnetworksUDPEstablished                  object
  serverDictClassBnetworksEstablished                     object
  clientDestinationPortNumberOfFlowsUDPEstablished        object
  clientDictNumberOfDistinctCountriesNotEstablished       object
  serverDictOfConnectionsTCPEstablished                   object
  serverDictNumberOfDistinctCountriesNotEstablished       object
  clientDictClassBnetworksUDPNotEstablished               object
  clientDestinationPortTotalBytesTCPEstablished           object
    80 : 699544                                           number
    443 : 188857                                          number
    5535 : 763                                            number
    18847 : 800                                           number
    37413 : 740                                           number
    55069 : 786                                           number
    58122 : 737                                           number
```

Figure 3.8: Raw data. This tab is for the viewing the data in the JSON file. This allows to check exact values for every feature. All data of all features are accessible here. The JSON structure of the profile file is visualized here.

3.4 Similarity between two profiles

To find out if two profiles are similar, we need to use similarity coefficients between selected features in the comparison of two profiles. Later we combine the values of the coefficients of all the features into one probability that describes the similarity of two profiles. This probability is the output of the classifier and the final coefficient that describes the similarity between two profiles.

3.4.1 Features used in the similarity of two profiles.

In the Section 3.2 several types of features are proposed. However for the similarity of two profiles we decided to focus on the Connection-based features (subsection 3.2.3) and the Port-based features (subsection 3.2.4). We believe that these two features groups represent most information. These features were selected because they can be understood(after normalization) as discrete probability distributions and therefore the probability distribution metrics can be used on them.

The Hour summary features 3.2.1 are not used in similarity measure because they aggregate information present in the original features. We do not use the Location-based features 3.2.2 as gathering this information takes too much time, and data from the Whois will be unavailable in Europe with the GDPR legislation [16]. The IP addresses-based features 3.2.5 are not used as we believe what the same information is present in the combination of the Connection features and the Port features and their format is not a histogram.

We decided to remove the features that are not related to the user behavior from the consideration. These features are all Port-based features describing source port of the connections initialized from the internet and the features describing destination port of the connection to the server where the connection was not established. From the Connection-based features the features that store information from the connections from the internet to the server where the connection was not established. These features user cannot influence by his behavior. However, these features can be used in the future by researchers for example for modeling attacks to user.

3.4.2 Similarity coefficient of two features

The Port features and the Connection features have the same structure. After normalization, they both represent discrete probability distributions. The structure of the Port-based features is always the port and value of the selected feature. The structure of the Connection-based features is an IP address or class B network and a value of the feature.

We used three methods for computing similarity between two discrete probability distributions:

- Bhattacharyya coefficient. Bhattacharyya coefficient measure the amount of overlapping between two distributions. Bhattacharyya coefficient(BC) [17] is defined as:

$$\sum_{i=1}^{n} \sqrt{p_i * q_i}$$

In this work we use reverted value of the coefficient so our value correspond to the

$$ARBC = 1 - BC$$

This means, that in this work the value 0 of the coefficient means match, 0.55 means half-match and the 1 means is mismatch.

- Chi-square [18]:

  $$\sum_{i=1}^{n} \frac{p_i^2 - q_i^2}{p_i + q_i}$$

  For the chi-square the exact match is 0,the half-match is 0.67 and 2 means mismatch.

- Intersection [18]:

  $$\sum_{i=1}^{n} min(p_i, q_i)$$

  For the intersection the exact match is 1,the half-match is 0.5 and mismatch is 0.

The probability distributions of the same feature in the two profiles are the p and q. The n is the number of partitions. p_i and q_i are the numbers of members of samples the p and q in the i-th partition. The size of n is the size of merged key sets(ports,IP addresses or class B networks) of two features. The example comparison of these three methods is shown in three situations.

- Comparison of the Class B network feature from profiles from distant dates in the dataset. The user and the hour is the same. In this case the 3 pm. This situation is visible in Figure 3.9
- Comparison of the Class B network feature from profiles from adjacent dates in the dataset. The user and the hour is the same. In this case, 10 am. This situation is visible in Figure 3.10
- Comparison of the Class B network feature from profiles from adjacent dates in the dataset. The user and the hour is the same. In this case, 4 am. In this situation, the user is not probably interacting with the machine. This situation is visible in Figure 3.10

These Figures are shown for getting an idea what coefficient value to expect based on the real example. From extensive testing, the Bhattacharyya coefficient was selected because it is fast to compute and brought best results in comparison to intersection and Chi-square method. But the difference between these methods is not too high, as all are based on per member value comparisons.

3.4.3 Similarity coefficient of two profiles

The Bhattacharyya coefficient measures how much similar two profiles are according to the features. For every single feature used, we calculate the value of the Bhattacharyya coefficient. That gives the same number of results as the number of features used when we compare two profiles. We want to classify the comparisons based on the values of the coefficients of each feature. We need to define groups that we want to classify for training:

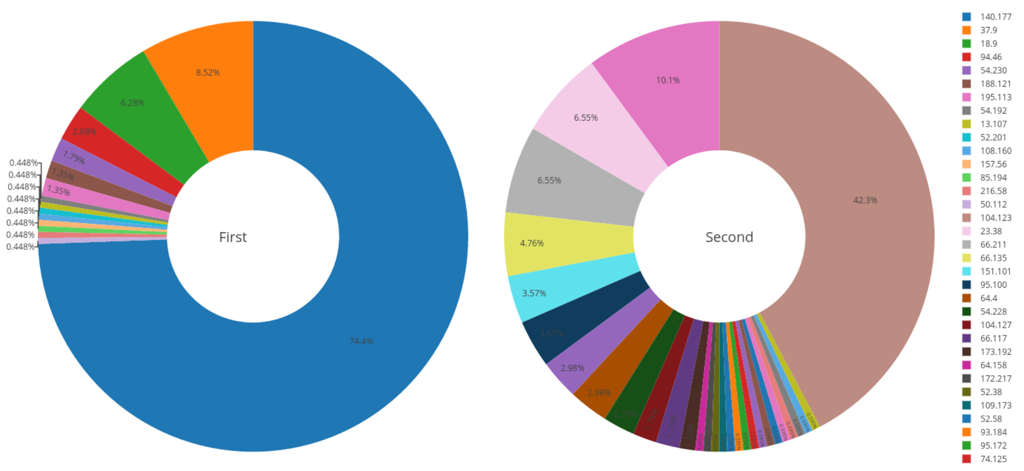


Figure 3.9: This Figure shows the comparison of feature "clientDictClassBnetworksTCPEstablished" (contacted class B networks) from two profiles from distant dates. The hour is 3 pm in both cases. The similarity between this classB feature according to the proposed methods is: Bhattacharyya: 0.863, Chi-square: 1.864, Intersection: 0.046. All three methods say that these two profiles are quite different. This unsimilarity corresponds to the data in this Figure. The exactly visited class B networks are not important, important is the overlap.

- The *similar* user behavior in the profiles comparison is defined as a comparison between two profiles from the same user from the same hour from different days.

- The *different* user behavior is defined as a comparison between two profiles from different users from any day.

This selection of comparisons means that profile from 10 a.m. is always compared to another profile from 10 am. The purpose of only comparing same daily hours is to catch the same type of user behavior. Profiles of the same users proved to be more stable in the similarity to themselves when we took profiles from different days and the same hour. The reason seems to be that the type of user behavior changes during the day. Comparing profiles from the next hours showed nicely events such as coming to work and leaving it as the computers stayed on. These events were visible as the significant spikes in the similarity of the profiles comparison. In Figure 3.13 there is an example of the comparison of the same user behavior between two days.

The result of the classifier is a probability that these two profiles are *similar* or *different*. We can view the resulting probability as a coefficient that tells us, how two profiles are close.

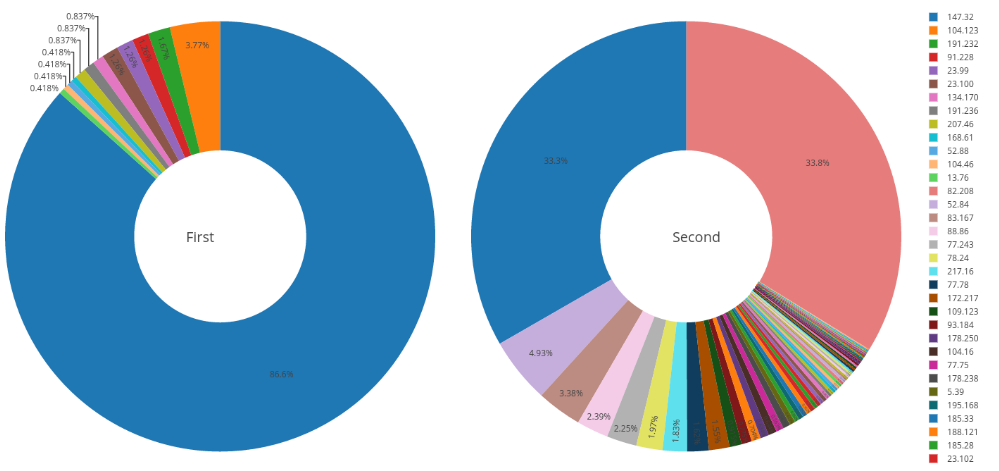


Figure 3.10: This Figure shows the comparison of feature "clientDictClassBnetworksTCPEstablished" (contacted class B networks) from two profiles from next dates. The hour is 10 am in both cases. The similarity between this classB features according to the proposed methods is: Bhattacharyya: 0.425, Chi-square: 0.982, Intersection: 0.350. All three methods show some similarity in this comparison. However, it is not clear from these values if this comparison is from the same or different user. The values correspond to this Figure that we can see some similarity there. The exactly visited class B networks are not important, important is the overlap.

3.5 Datasets for profile to profile classifier

We define the profiles from September 2016 as dataset A. We define the profiles from 1st fo 6th October 2016 as dataset B. Diagram explaining datasets is shown in Figure 3.12. The number of profiles per users IP addresses in dataset A is described in Table 3.2. The number of profiles per users IP addresses in dataset B is shown in Table 3.3.

The comparisons of the *similar* user behavior are labeled as 0. The comparisons of the *different* user behavior are labeled as 1.

As there is usually much more comparisons of the *different* user behavior, the downsampling is used to balance the dataset.

For the training and cross-validation, the dataset A is used. There are 261,571 samples of *different* user behavior (label 1) and 61,500 samples of *similar* user behavior (label 0). By using downsampling, we settle in 61,500 samples from each class. So, in the end, the classifier learns over 123,000 samples. For the testing, the dataset B is used. The testing data had 4876 samples in total.

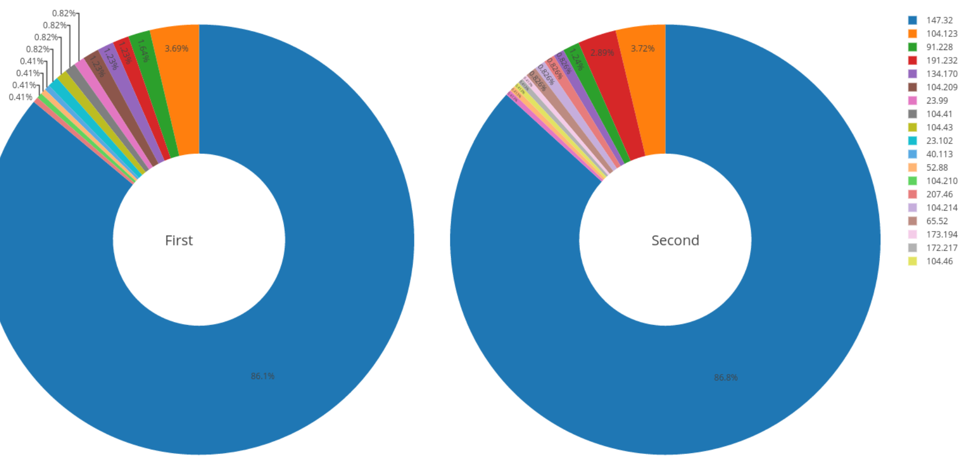


Figure 3.11: This Figure shows the comparison of feature "clientDictClassBnetworksTCPEstablished" (contacted class B networks) from two profiles from next dates. The hour is 4 am in both cases. This profile is probably the comparison of the machine when the user is not there. The similarity between this classB features according to the proposed methods is: Bhattacharyya: 0.040, Chi-square: 0.085, Intersection: 0.943. All three methods tell us that these two profiles are very similar. The similarity is also well visible from this Figure. The exactly visited class B networks are not important, important is the overlap.

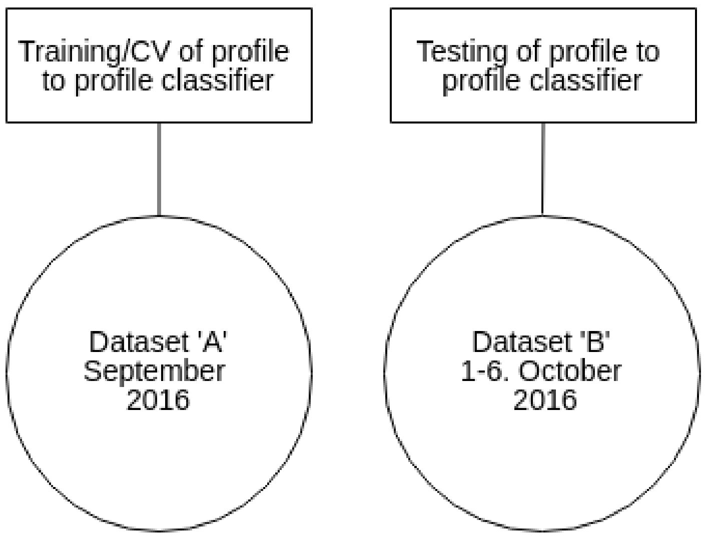

Training/CV of profile to profile classifier
Testing of profile to profile classifier
Dataset 'A' September 2016
Dataset 'B' 1-6. October 2016

Figure 3.12: The diagram describing the data used in profile to profile classifier.

IP	profiles	IP	profiles	IP	profiles
147.32.83.102	652	147.32.83.139	139	147.32.83.146	652
147.32.83.154	275	147.32.83.173	156	147.32.83.178	8
147.32.83.186	14	147.32.83.188	172	147.32.83.196	184
147.32.83.205	72	147.32.83.206	3	147.32.83.209	48
147.32.83.224	15	147.32.83.248	479	147.32.83.29	652
147.32.83.34	61	147.32.83.37	652	147.32.83.38	652
147.32.83.69	652				

Table 3.2: Number of profiles in dataset A by IP addresses.

IP	profiles	IP	profiles	IP	profiles
147.32.83.102	117	147.32.83.139	23	147.32.83.146	132
147.32.83.154	132	147.32.83.173	24	147.32.83.178	2
147.32.83.186	0	147.32.83.188	33	147.32.83.196	97
147.32.83.205	0	147.32.83.206	6	147.32.83.209	0
147.32.83.224	0	147.32.83.248	100	147.32.83.29	132
147.32.83.34	29	147.32.83.37	132	147.32.83.38	132
147.32.83.69	132				

Table 3.3: Number of profiles in dataset B by IP addresses.

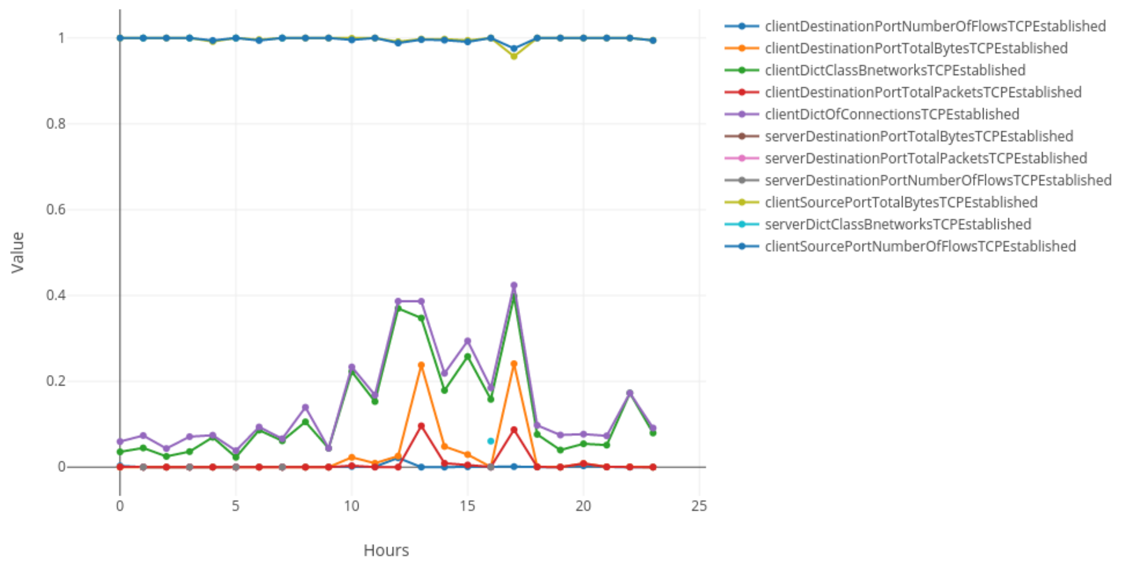


Figure 3.13: This Figure shows how much is a user similar to the same hour in next days.

3.6 Profile to Profile classifier

In this Section, we present three different classifiers used on the dataset presented in Section 3.5. We compare Decision tree, Random forest, and XGBoost. To train the classifiers we used the sklearn toolkit [19] and XBoost library [20]. We selected the tree family of classifiers because of their ability to show the importance of the features. This is useful for analyzing why the classifier considers two profiles similar or not.

3.6.1 Decision tree

The Decision tree classifier learns simple decision rules inferred from the features in the data. The Decision tree served as a simple classifier to get some baseline results. The resulting decision tree is visible in Figure 3.14. The tree is limited to the maximal depth of three to make it simple and easily visualized. The other parameters of the Decision tree are the default ones used in the sklearn toolkit [19].

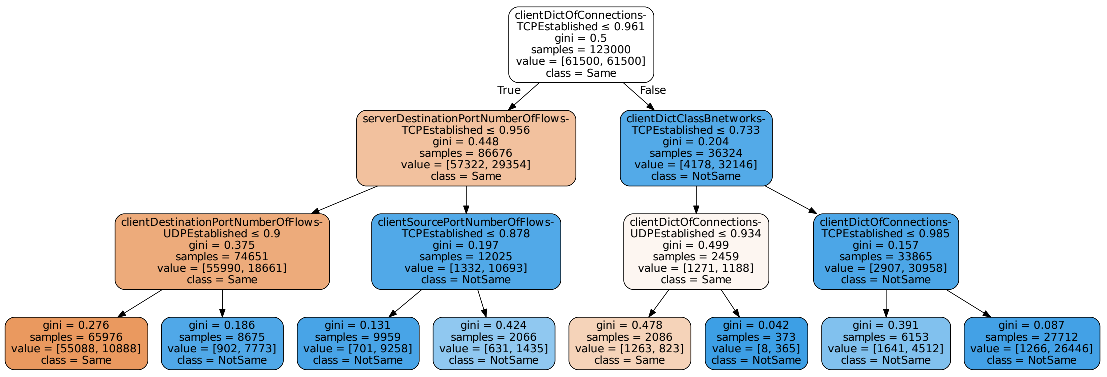

clientDictOfConnections-TCPEstablished ≤ 0.961 gini = 0.5 samples = 123000 value = [61500, 61500] class = Same
True
False
serverDestinationPortNumberOfFlows-TCPEstablished ≤ 0.956 gini = 0.448 samples = 86676 value = [57322, 29354] class = Same
clientDictClassBnetworks-TCPEstablished ≤ 0.733 gini = 0.204 samples = 36324 value = [4178, 32146] class = NotSame
clientDestinationPortNumberOfFlows-UDPEstablished ≤ 0.9 gini = 0.375 samples = 74651 value = [55990, 18661] class = Same
clientSourcePortNumberOfFlows-TCPEstablished ≤ 0.878 gini = 0.197 samples = 12025 value = [1332, 10693] class = NotSame
clientDictOfConnections-UDPEstablished ≤ 0.934 gini = 0.499 samples = 2459 value = [1271, 1188] class = Same
clientDictOfConnections-TCPEstablished ≤ 0.985 gini = 0.157 samples = 33865 value = [2907, 30958] class = NotSame
gini = 0.276 samples = 65976 value = [55088, 10888] class = Same
gini = 0.186 samples = 8675 value = [902, 7773] class = NotSame
gini = 0.131 samples = 9959 value = [701, 9258] class = NotSame
gini = 0.424 samples = 2066 value = [631, 1435] class = NotSame
gini = 0.478 samples = 2086 value = [1263, 823] class = Same
gini = 0.042 samples = 373 value = [8, 365] class = NotSame
gini = 0.391 samples = 6153 value = [1641, 4512] class = NotSame
gini = 0.087 samples = 27712 value = [1266, 26446] class = NotSame

Figure 3.14: The decision tree used in the comparison of the classifiers.

3.6.2 Random Forest

Random forests are an ensemble learning method for classification. They are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [21]. A random forest is a meta estimator that fits many decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The implementation of the Random forest is from the sklearn toolkit [19].

The important parameters of the random forest classifier are:

- max_depth is the maximum depth of a tree.
- n_estimators is the number of trees to fit.
- max_features refers to the size of the random subsets of features to consider when splitting a node.

- min_samples_split is the minimum number of samples required to split an internal node.

Random forest parameters All the parameters were established using grid search and cross-validated by using five folds cross-validation. The best performing parameters proved to be:

- max_depth=35
- n_estimators = 1000
- max_features = 7
- min_samples_split = 2

3.6.3 XGBoost

The usually best performing classifier from this family of classifiers is the XGBoost(Extreme Gradient Boosting) classifier [22]. It is the Distributed Gradient Boosting tree classifier. The booster used is the booster with logistic regression for binary classification. The important XGBoost parameters are:

- max_depth is the maximum depth of a tree.
- min_child_weight defines the minimum sum of weights of all observations required in a child. We would expect, that the value of min_child_weight would be higher after cross-validation than 1, but this value showed best results.
- learning_rate makes the model more robust by shrinking the weights on each step. Resulting value 0.1 was expected.
- n_estimators is the number of boosted trees to fit.
- gamma makes the algorithm conservative. Gamma specifies the minimum loss reduction required to make a split.

XGBoost parameters All the parameters were established using grid search and cross-validated by using five folds cross-validation. The best performing parameters proved to be:

- max_depth = 18
- min_child_weight = 3
- learning_rate = 0.1
- n_estimators = 151
- gamma = 0.4

The implementation of the XGBoost is from the XGBoost library [20] and the sklearn API is used [19].

The code used for the cross-validation and tuning of the parameters and their explanation is based on this great online article [23].

3.7 Experiment results

In this section, we show the resulting importance of the features and the overall results of the profile to profile classifiers.

3.7.1 Feature importances

As we use the tree types of classifiers, we can see which features are mainly used for the classification. We measure the Feature importance score, feature importance score simply sums up how many times each feature is used for a split in tree estimators and normalize it.

- Decision tree - The Feature importance score of each feature is visible in the Figure 3.15. These features correspond to the tree itself in the Figure 3.14.
- Random forest - The Feature importance score of each feature is visible in the Figure 3.16. Client features dominate, but some of the server features are important.
- XGBoost - The Feature importance score of each feature is visible in the Figure 3.17. We can see that client based features are dominant in our dataset as our dataset consisted of the users. If our dataset consists only the servers, we would expect server features to be most important. Howe some server features are taken into consideration as some of the users use some p2p traffic.

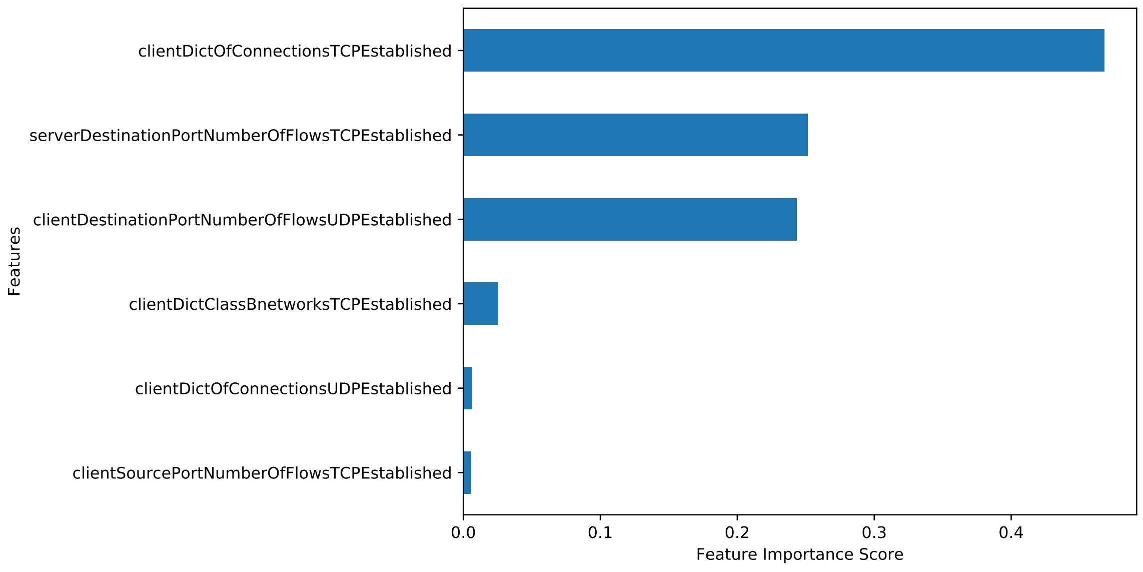


Figure 3.15: Feature importance of the features of the Decision tree classifier. The first six best performing features are shown in this Figure.

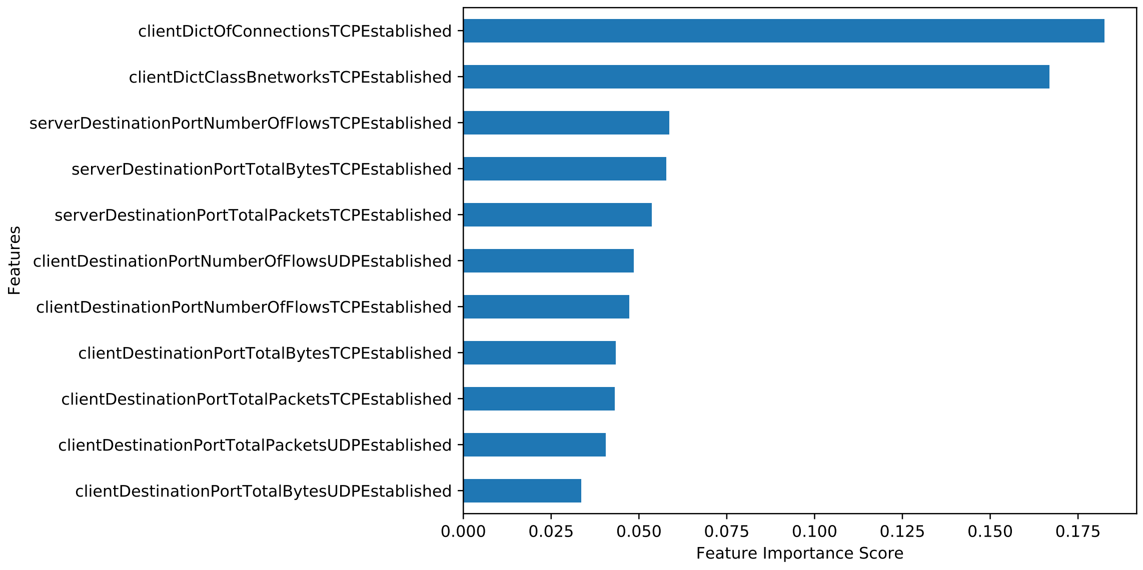


Figure 3.16: Feature importance of the features in the Random forest classifier. The first 11 best performing features are shown in this Figure.

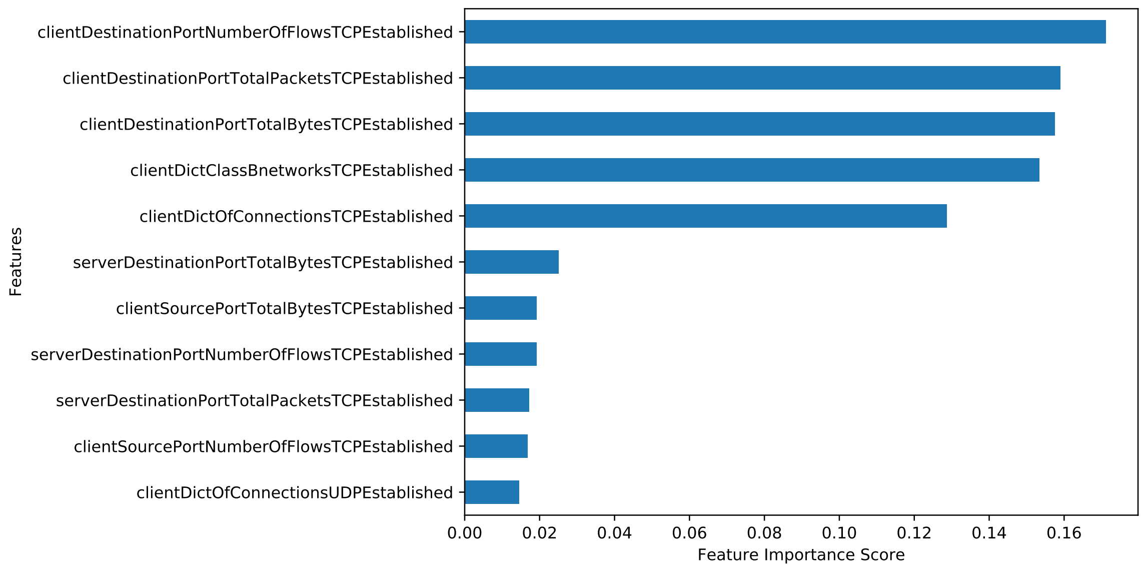


Figure 3.17: Feature importance of the features in the XGBoost classifier. The first 11 best performing features are shown in this Figure. The client features dominates the decision.

3.7.2 Classifiers results

The five folds cross-validation was used on the training data for parameter estimation. The results of all classifiers on training data, testing data and averaged results from cross-validation of final parameters are shown in Table 3.4. The confusion matrix of the Decision tree classifier on the testing data is in Figure 3.18, of the Random forest classifier, is in Figure 3.19 and of the XGBoost classifier in Figure 3.20.

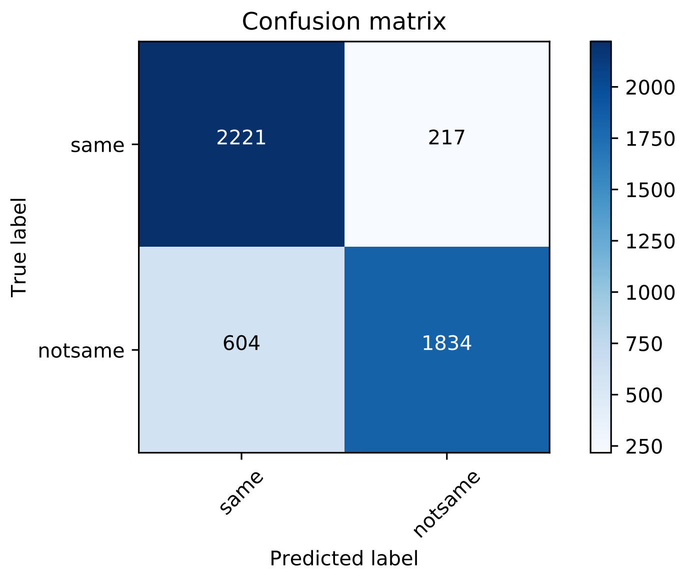


Figure 3.18: Confusion matrix of the Decision tree classifier on the testing data.

3.8 Discussion

The results of the Decision tree are quite impressive as it is an elementary Decision tree. The Random forest and the XGBoost perform very similarly, but Random forest slightly outperforms the XGBoost. The drop in the accuracy and higher False-Positive rate on the testing data of the Random forest and the XGBoost can be caused by the changed behavior of the users in October or simply by over-fitting on the training data.

	Decision tree			Random forest			XGBoost		
metric	TR	CV	Test	TR	CV	Test	TR	CV	Test
accuracy	0.86	0.86	0.83	0.97	0.91	0.86	0.95	0.91	0.85
roc_auc	0.88	0.88	0.84	0.99	0.98	0.95	0.99	0.98	0.94
f1	0.86	0.85	0.82	0.97	0.91	0.86	0.95	0.91	0.85
precision/PPV	0.91	0.91	0.89	0.96	0.92	0.84	0.94	0.92	0.83
recall/TPR	0.81	0.81	0.75	0.97	0.91	0.88	0.95	0.91	0.88
FPR	0.08	0.08	0.09	0.04	0.08	0.16	0.06	0.08	0.18

Table 3.4: The results of the profile to profile classifiers.

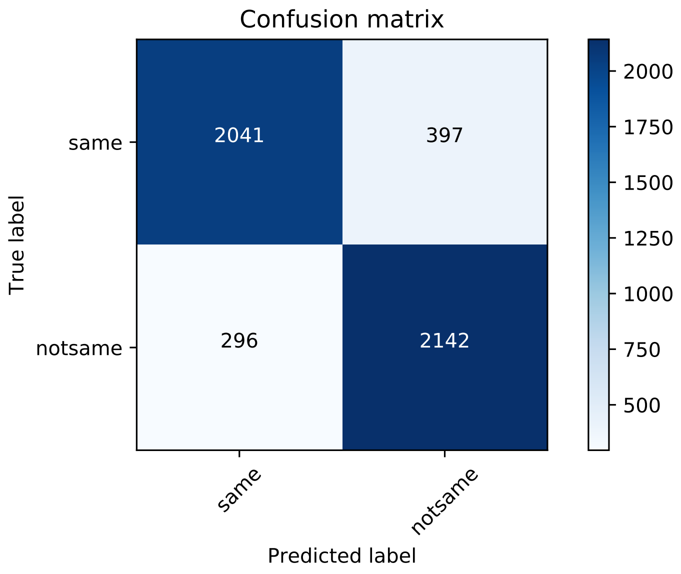


Figure 3.19: Confusion matrix of the Random forest classifier on the testing data.

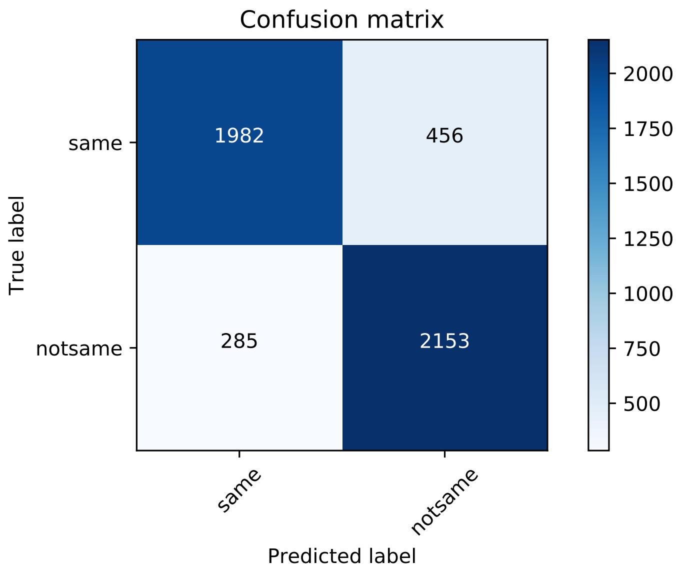


Figure 3.20: Confusion matrix of the XGBoost classifier on the testing data.

3.9 Conclusion of individual profiles classification

We presented a way how to create time-based profiles from the *binetflows* capture. Time-based profiles allowed us to analyze larger captures thanks to the visualization of the profiles. We observed the similarity between the profiles and developed a per features similarity coefficients. We designed the profile to profile classifiers based on these similarity coefficients with accuracy more than 83%. The best performing classifier has accuracy of 86%. The resulting probability of the classifier gives us information how much the compared profiles are similar. The comparison of the two profiles now can give us an idea of if these two profiles come from the same user. Combining multiple profile comparison is presented in next Chapter 4.

Chapter 4

User detection

In previous Chapter 3 we developed the profile to profile classifier that managed to have an accuracy of 86%. Although good, these results are less useful when we have to compare several profiles simultaneously. This problem was the motivation to work on a second classifier, capable of classifying the complete traffic of a user. To achieve this task we combined the information from the comparison of several profiles and took a final decision for the user being analyzed. The comparison of users to find similar behaviors and ultimately similar users is a difficult task. The main problem is that the decisions about a user have to be aggregated somehow, and it means that some information has to be compromised. In this Chapter, we describe the method we used to compare two users and to find out whether they are the same or not. We start with the used dataset, than we continue with method implementation and we finish with the results.

4.1 Dataset for User Detection

In order to compare users with users we need a dataset that consists of many profiles per user. The dataset used in this thesis, as it was described in Section 3.5, is composed of more than one and half month of real users data from an University network. The most important characteristic of our dataset is that the users were verified, meaning that we know them. This verification is the reason why we have 19 users to work with.

For the training of the threshold classifier we use datasets A and B, as they were described previously. For testing the threshold classifier we defined a new dataset C that consists of user traffic from Oct 13th, 2016, to Oct 15th, 2016. However, since we need to compare two sets of users in order to obtain results, the testing is done comparing the new dataset C against the already described dataset A. This comparison obtains the final results for our user-to-user classifier. Each comparison in the user-to-user classifier is done with the profile to profile classifier from Chapter 3. The relation between datasets is visible in Figure 4.1.

There are 19 users in dataset A and 15 users in dataset B and 15 users C. In order to be clear with the description of the dataset we have to say that the same IP address may appear in datasets A, B and C, however for the purpose of the training and testing we take their traffic as separated. However, we still consider the IP the be coming from the same user in different days. The 15 users from dataset B and C also appear in dataset A, four

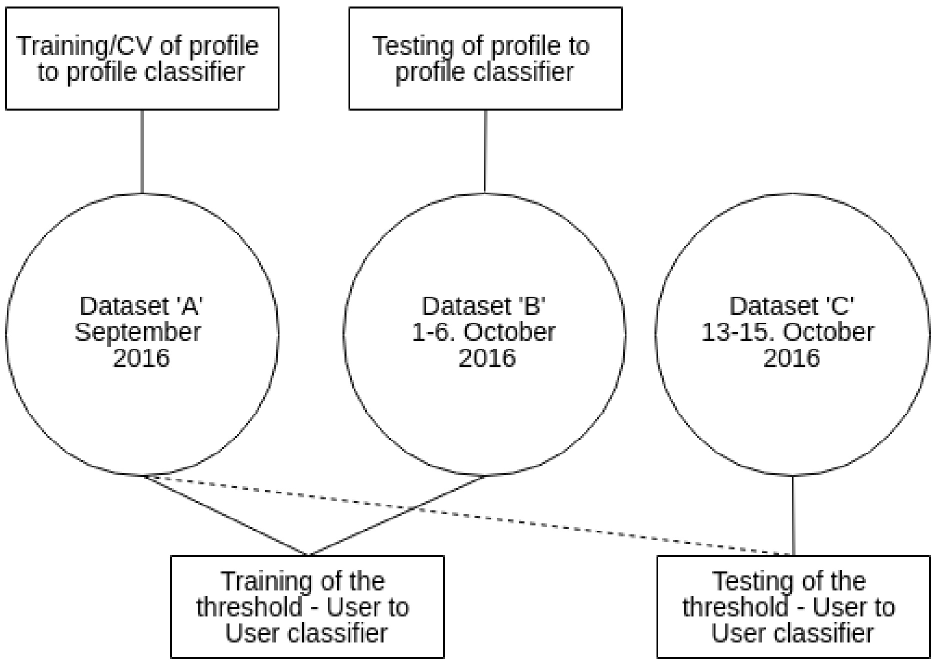

Training/CV of profile to profile classifier
Testing of profile to profile classifier
Dataset 'A' September 2016
Dataset 'B' 1-6. October 2016
Dataset 'C' 13-15. October 2016
Training of the threshold - User to User classifier
Testing of the threshold - User to User classifier

Figure 4.1: This diagram describes the relationship between the datasets used in the profile to profile classifier and the datasets used for user classification. We train the threshold using the dataset A and B. We use the dataset C for testing. However, because the profile to profile classifier from Chapter 3 is trained on the dataset A, we test with comparing users from the dataset C to the users in the dataset A.

of the users in dataset A do not have any data in datasets B and C. There are 28 days in dataset A, 6 days in dataset B and 3 days in dataset C. The number of profiles for each user in dataset A is described in Table 3.2. The number of profiles for each user in dataset B is described in Table 3.3. The number of profiles for user in dataset C is described in Table 4.1.

4.2 Users Detection Method

In order to compare and detect users by means of a classifier it is necessary to design a proper method. Our method is based on the idea of aggregating several profile to profile comparisons in one final decision. The method can be summarized as:

1. Take all the profiles of a given testing user U_2, the user we want to find among users in the dataset A.
2. Take all the profiles of a given trained user U_1, the user that belongs to the dataset A.
3. For each profile of U_2 do:
    (a) Extract the hour when this profile was created.
    (b) Find all the profiles of the user U_1 for the same hour. There is at most 1 profile per day.

IP	profiles	IP	profiles	IP	profiles
147.32.83.102	60	147.32.83.139	15	147.32.83.146	60
147.32.83.154	30	147.32.83.173	16	147.32.83.178	0
147.32.83.186	2	147.32.83.188	60	147.32.83.196	60
147.32.83.205	0	147.32.83.206	0	147.32.83.209	0
147.32.83.224	12	147.32.83.248	40	147.32.83.29	60
147.32.83.34	6	147.32.83.37	60	147.32.83.38	60
147.32.83.69	61				

Table 4.1: Number of profiles in dataset C by IP addresses.

   (c) Compute the profile-to-profile probability between the two profiles with the best classifier of Section Chapter 3

   (d) Store the average of these probabilities.

4. Use the trained threshold (described later) to decide if the two users are the same or not.

An important characteristic of the user to user classifier is that each profile is compared using the pre-trained model from the profile to profile comparison described in Chapter 3. Our classification method is simple enough. But given our dataset and out training process its possible to have a good and simple linear decision on the output.

4.2.1 Training the threshold selection

For the training we use datasets A and B. Comparisons for training the threshold between dataset B and A are visible in the heatmap in Figure 4.4. The number of the comparisons in the heatmap is visible in Figure 4.6.

The training phase of the user to user classification method has the goal of finding the best threshold. The result of every comparison between two profiles is a value between 0 and 1. If the value is closer to 0, it means that it is more probable that this comparison came from the same user. By averaging all the comparisons for all the profiles, we get one value between 0 and 1 representing the probability of these two users being the same. To be able to decide if the two users are similar we train the simple threshold-based classifier.

To the user in dataset B, we always match one user from dataset A. We decided that the users with the lowest resulting value, here the user with the highest probability, are matched. However, the minimum is not always correct. There are situations where the user is not similar to himself leading to the mismatch. Taking the only minimum would then cause wrong user identification. To avoid these false defections we decided to create a threshold, only resulting values below this threshold leads to the user match, otherwise the user is not matched at all.

4.2.2 Establishing threshold value and results on the training data

We selected the threshold based on the heatmap in the Figure 4.4, it is visible that last threshold that does not have any False-Positive on the training data is threshold **0.32**.

Any higher threshold would mean that the IP address 147.32.83.34 would be identified as IP address 147.32.83.69. The table showing the matched users from the heatmap in the Figure 4.4 is in Figure 4.5.

4.2.3 Evaluaton metrics

Given testing user U_2, the user we want to find among users in the dataset A. The values of True-Positives, False-Positives, True-Negatives and False-Negatives represents:

- The True-Positive value represents the situation where the minimum value of the comparison between a user U_2 and the users from the dataset A is under the threshold. Also, we match the user U_2 correctly to the corresponding user from the dataset A.
- The False-Positive value represents the situation where the minimum value of the comparison between a user U_2 and the users from the dataset A is under the threshold. Also, we did not match the user U_2 to the corresponding user from the dataset A.
- The True-Negative value represents the situation where the minimum value of the comparisons between a user U_2 is higher than a threshold, and we match correctly that the user U_2 is not any of the users from the dataset A.
- The False-Negative value represents the situation where the minimum value of the comparisons between a user U_2 is higher than a threshold, and we match incorrectly that the user U_2 is not any of the users from the dataset A.

4.2.4 Training results

The confusion matrix using the threshold **0.32** is in Figure 4.2. Results are

Accuracy	0.6
F1	0.75
FPR	Not defined
TPR	0.6
Precision	1
FNR	0.4

That means that from 15 users in dataset B we correctly matched 9 and were not able to match 6.

4.3 Testing Experiments

Based on the fact that we have a comprehensive dataset of users, it is possible to use the trained method described in previous section for the testing experiments. As it was described in Section 4.1, the dataset for testing consist in our dataset C. However, our method needs two datasets for finding a probability and therefore we compare our *new* users from the dataset C against our *known* users in dataset A.

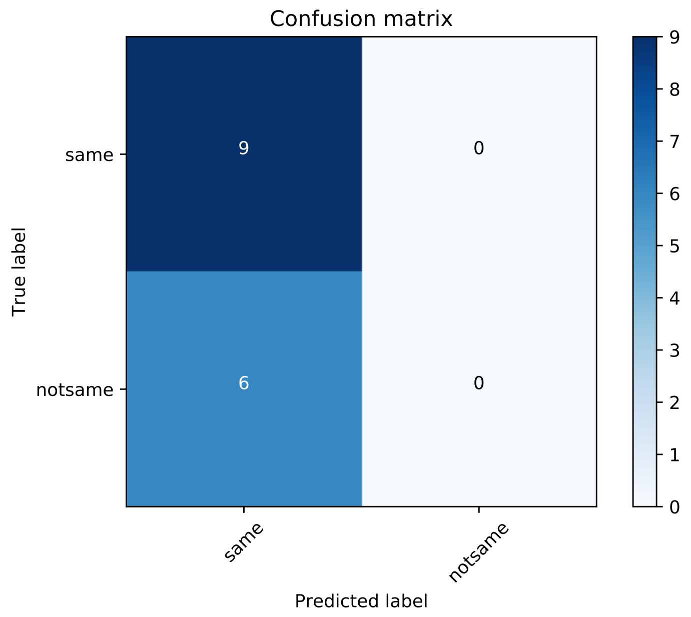


Figure 4.2: Confusion matrix of the training of the user to user classifier. Meaning of the TP,TN,FN and FP is described in subsection 4.2.3.

The testing experiments consists in applying our detection method, as it was described in Subsection 4.2, to compare all the profiles of the users in dataset C, against our *known* users from dataset A.

The results values for the testing comparisons are described in the Figure 4.7. This Figure shows the results of comparing the users in dataset C against the users of dataset A. For each user we average the profile-to-profile probabilities of all the comparisons and we use the threshold to decide if they are considered the same user or not. The colors show the values of the averaged probabilities. Values closer to 0 (darker) means more similar.

Since the comparison between two users involves different amount of data, we show in Figure 4.9 the amount of comparisons done between two users of the testing phase. Every cell has at most 3*28*24 comparisons: 3 is the number of days in dataset C, 28 is the number of days in dataset A, and 24 is the number of hours in the day. In total there can be a maximum of 2016 comparison for two users. This Figure does not show decisions, but amount of comparisons.

After using the threshold of 0.32 for classifying users, Figure 4.8 shows the final decision of which user is detected as which. This Figure has a black box the the users are considered the same, and a color box when there is an error.

With these comparisons for the testing phase, we can now compute the results and performance metrics. All the comparisons use the previously trained threshold of **0.32**: The confusion matrix is in Figure 4.3. Results are

Accuracy	0.6
F1	0.75
FPR	1.0
TPR	0.643
Precision	0.9
FNR	0.357

These results mean that from 15 users in dataset C we correctly matched 9 and the method was not able to find 5 users. It also means that only one user was incorrectly identified. Unfortunately, the lack of more users (despite having traffic from 45 days) means that our FPR is 100%. However, this value is not representative of what the algorithm can do. We consider these results very good, because its know that the behaviors of some users change some much that its not possible to detect them precisely.

The source code for training the classifier and for running the experiments can be found in the GitHub repository [24].

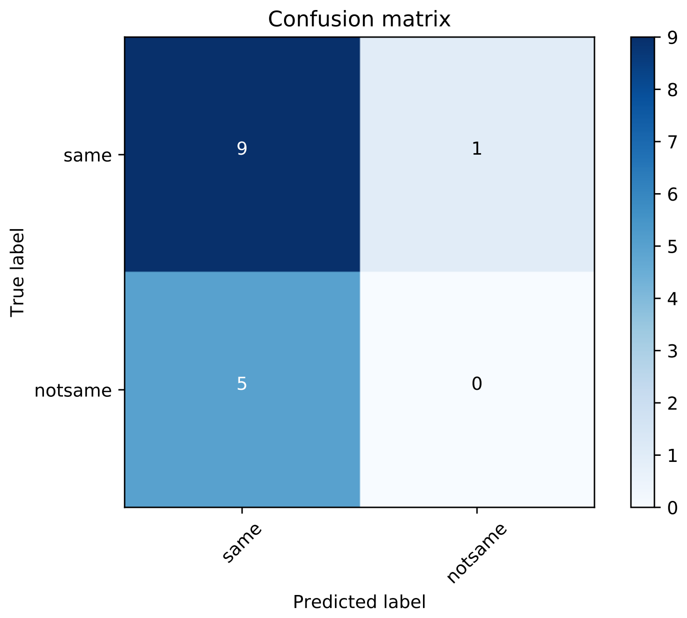


Figure 4.3: Confusion matrix of the testing of the user to user classifier. Meaning of the TP,TN,FN and FP is described in subseection 4.2.3.

4.4 Analysis of Results

We were able to correctly match most of the users. However, one user was matched incorrectly. We lost some important details by heavily averaging the profile comparisons. For example, the last day of the user with IP addresses 147.32.83.154 of the dataset A is super similar to days in dataset B. Visible in Figure 4.10. But as the most of the month is not similar, the result is 0.52.

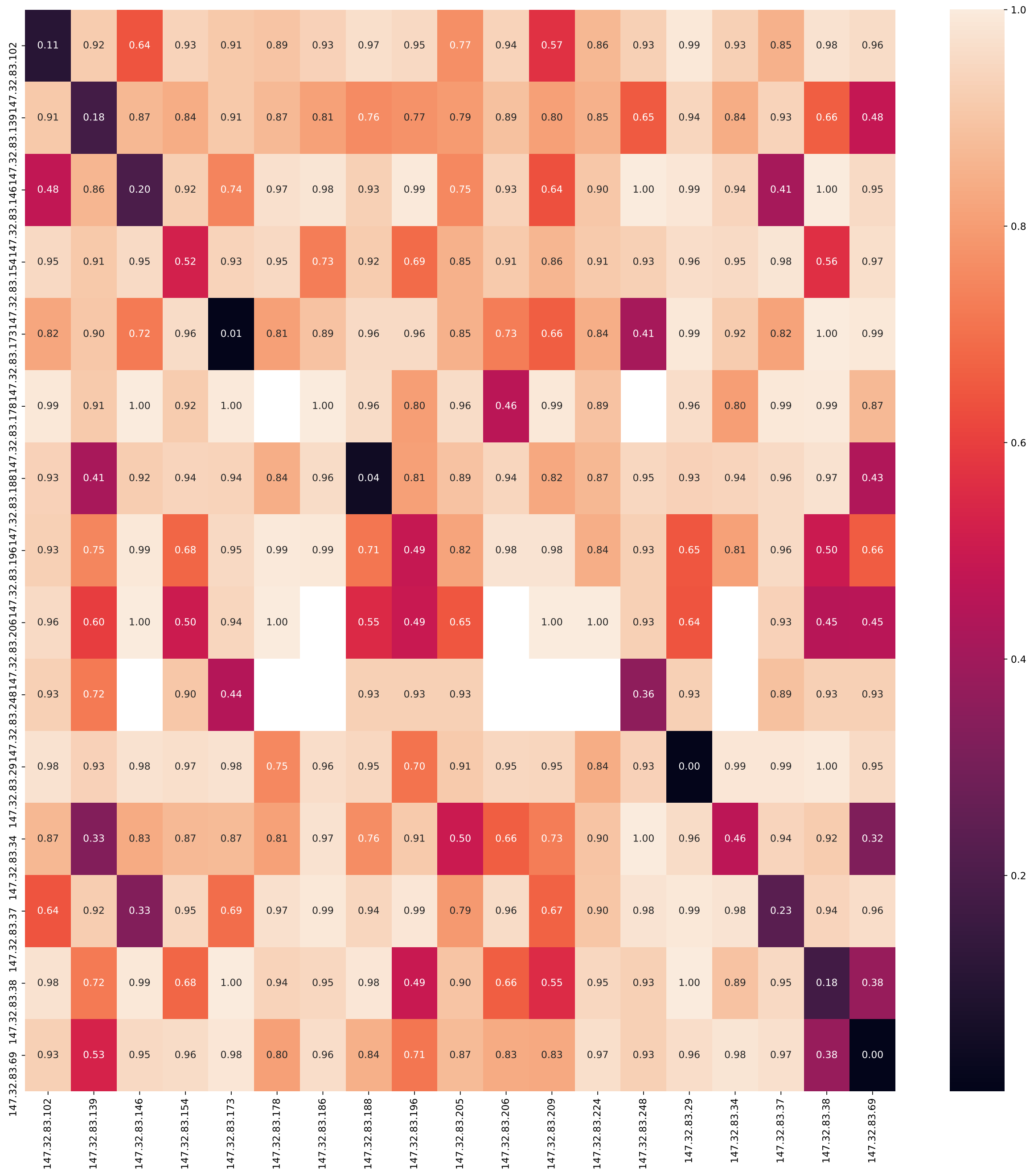


Figure 4.4: This heatmap shows relations between users. The rows represent IP addresses from the dataset B. The columns represent IP addresses from the dataset A. Four IP addresses are not present in the dataset B. The white spaces are places there were no data for comparisons. For example, one user having only TCP and the second user only UDP traffic. The number of comparisons forming the decision is visible in Figure 4.6.

This raises the question if analyzing month capture is not too much. Maybe the solution is some better method than averaging can be used as it is discussed in Future work.

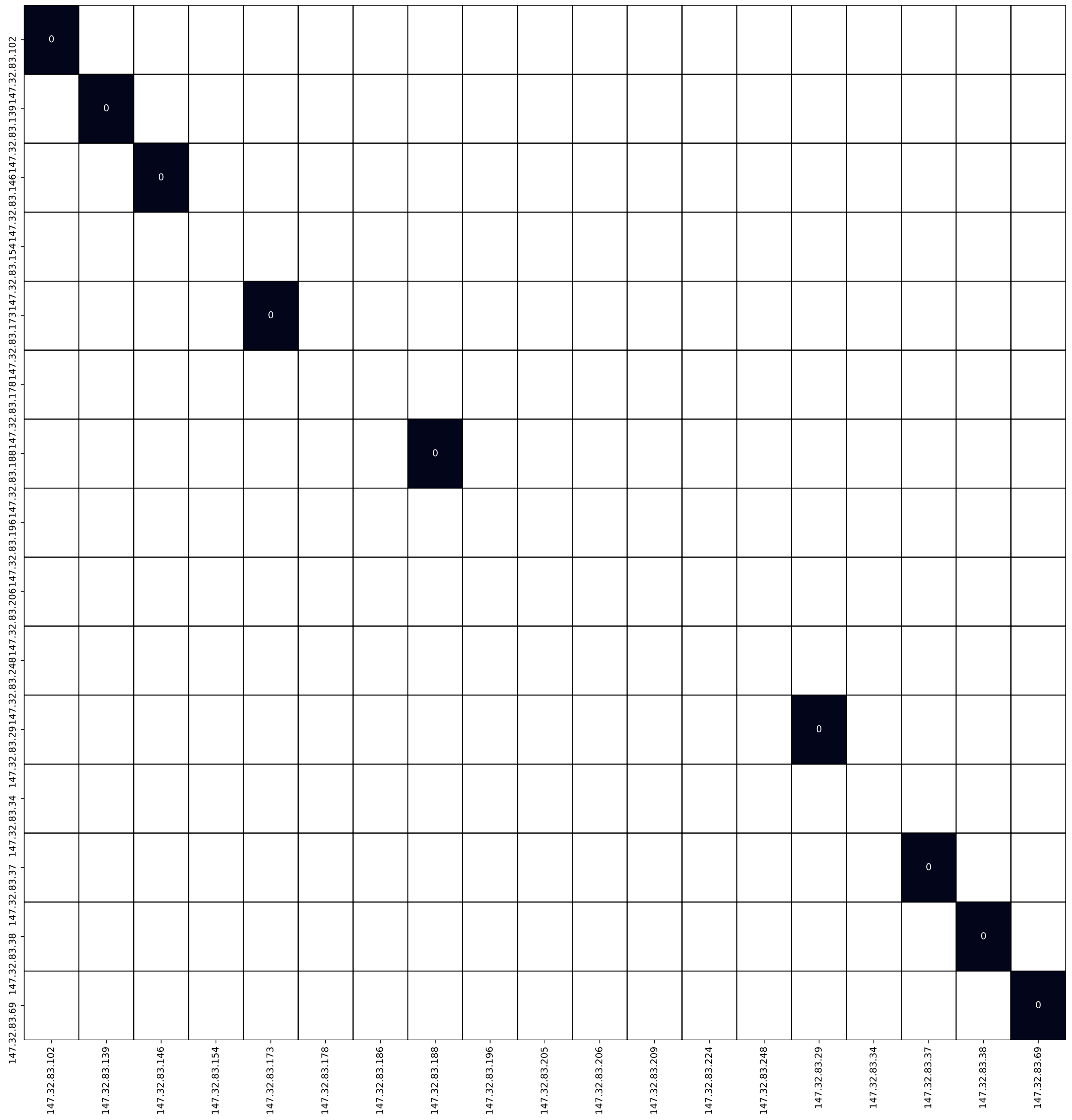
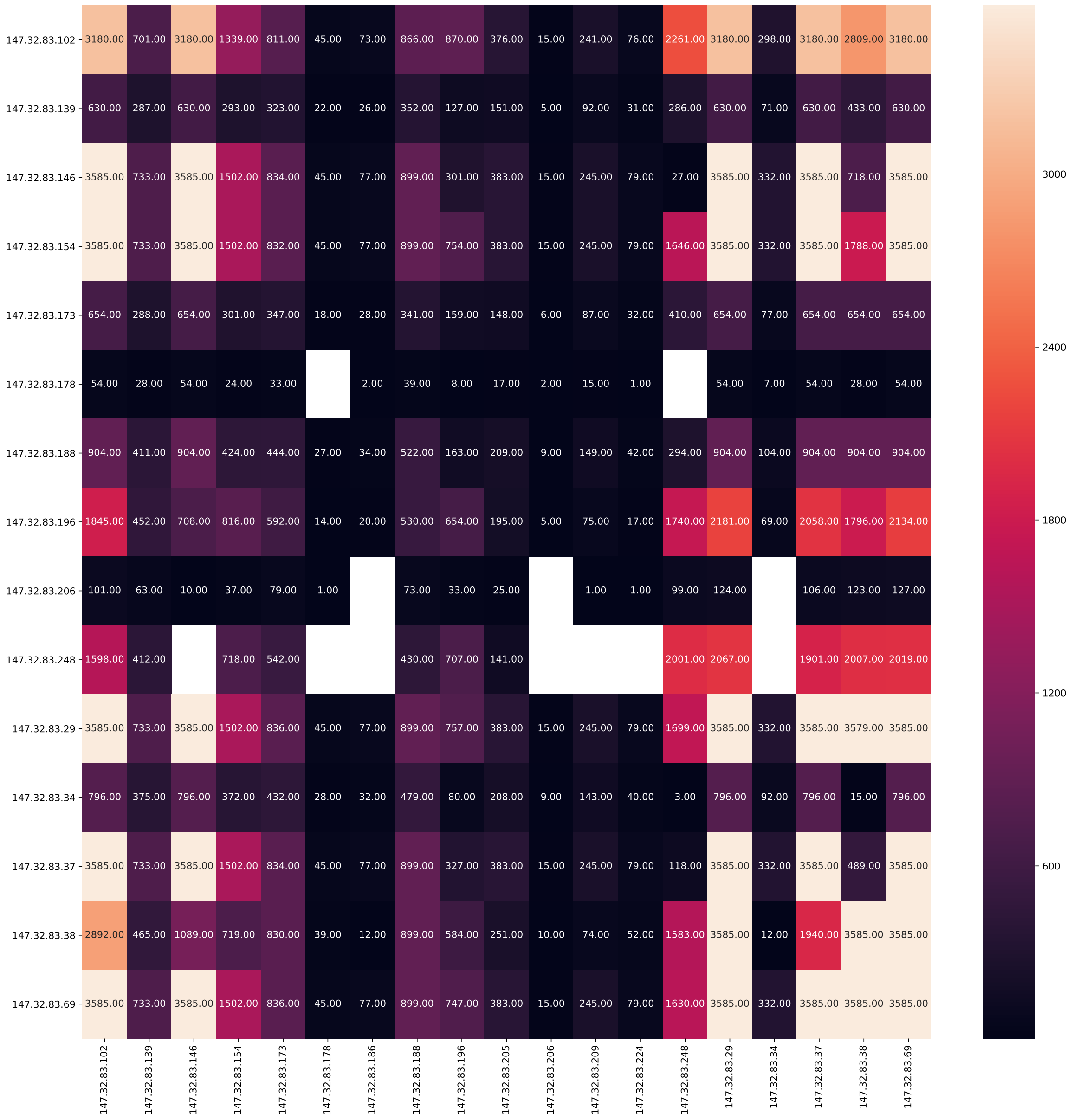


Figure 4.5: This Figure shows the values under the threshold value of 0.32 on the training data. The results are based on values from the heatmap in Figure 4.4. The zeros show the correctly matched users. Everything else has a higher value than the threshold. The users from dataset B are in the rows of the heat map. The users from dataset A are in the columns.

Even when we checked the users IP addresses several times; there was still chance that the users IP address changed. When the user changes IP address on the training data of the profile to profile classifier, it corrupts the result of this classifier. This corruption than propagates to the user to user classifier. The disadvantage of this method is that we need to be sure that change of the users IP address does not happen.

Creating the heatmap in Figure 4.4 and the heatmap in Figure 4.7 takes around 30



	147.32.83.102	147.32.83.139	147.32.83.146	147.32.83.154	147.32.83.173	147.32.83.178	147.32.83.186	147.32.83.188	147.32.83.196	147.32.83.205	147.32.83.206	147.32.83.209	147.32.83.224	147.32.83.248	147.32.83.29	147.32.83.34	147.32.83.37	147.32.83.38	147.32.83.69
147.32.83.102	3180.00	701.00	3180.00	1339.00	811.00	45.00	73.00	866.00	870.00	376.00	15.00	241.00	76.00	2261.00	3180.00	298.00	3180.00	2809.00	3180.00
147.32.83.139	630.00	287.00	630.00	293.00	323.00	22.00	26.00	352.00	127.00	151.00	5.00	92.00	31.00	286.00	630.00	71.00	630.00	433.00	630.00
147.32.83.146	3585.00	733.00	3585.00	1502.00	834.00	45.00	77.00	899.00	301.00	383.00	15.00	245.00	79.00	27.00	3585.00	332.00	3585.00	718.00	3585.00
147.32.83.154	3585.00	733.00	3585.00	1502.00	832.00	45.00	77.00	899.00	754.00	383.00	15.00	245.00	79.00	1646.00	3585.00	332.00	3585.00	1788.00	3585.00
147.32.83.173	654.00	288.00	654.00	301.00	347.00	18.00	28.00	341.00	159.00	148.00	6.00	87.00	32.00	410.00	654.00	77.00	654.00	654.00	654.00
147.32.83.178	54.00	28.00	54.00	24.00	33.00		2.00	39.00	8.00	17.00	2.00	15.00	1.00		54.00	7.00	54.00	28.00	54.00
147.32.83.188	904.00	411.00	904.00	424.00	444.00	27.00	34.00	522.00	163.00	209.00	9.00	149.00	42.00	294.00	904.00	104.00	904.00	904.00	904.00
147.32.83.196	1845.00	452.00	708.00	816.00	592.00	14.00	20.00	530.00	654.00	195.00	5.00	75.00	17.00	1740.00	2181.00	69.00	2058.00	1796.00	2134.00
147.32.83.206	101.00	63.00	10.00	37.00	79.00	1.00		73.00	33.00	25.00		1.00	1.00	99.00	124.00		106.00	123.00	127.00
147.32.83.248	1598.00	412.00		718.00	542.00			430.00	707.00	141.00				2001.00	2067.00		1901.00	2007.00	2019.00
147.32.83.29	3585.00	733.00	3585.00	1502.00	836.00	45.00	77.00	899.00	757.00	383.00	15.00	245.00	79.00	1699.00	3585.00	332.00	3585.00	3579.00	3585.00
147.32.83.34	796.00	375.00	796.00	372.00	432.00	28.00	32.00	479.00	80.00	208.00	9.00	143.00	40.00	3.00	796.00	92.00	796.00	15.00	796.00
147.32.83.37	3585.00	733.00	3585.00	1502.00	834.00	45.00	77.00	899.00	327.00	383.00	15.00	245.00	79.00	118.00	3585.00	332.00	3585.00	489.00	3585.00
147.32.83.38	2892.00	465.00	1089.00	719.00	830.00	39.00	12.00	899.00	584.00	251.00	10.00	74.00	52.00	1583.00	3585.00	12.00	1940.00	3585.00	3585.00
147.32.83.69	3585.00	733.00	3585.00	1502.00	836.00	45.00	77.00	899.00	747.00	383.00	15.00	245.00	79.00	1630.00	3585.00	332.00	3585.00	3585.00	3585.00

Figure 4.6: The heat map shows the sum of all comparisons between the dataset B and the dataset A. Every cell has at most 6*28*24 comparisons. Six is the number of days in dataset B. 28 is the number of days in dataset B. 24 is the number of hours in the day. In total 4032.

minutes on a machine with 22 CPUs.

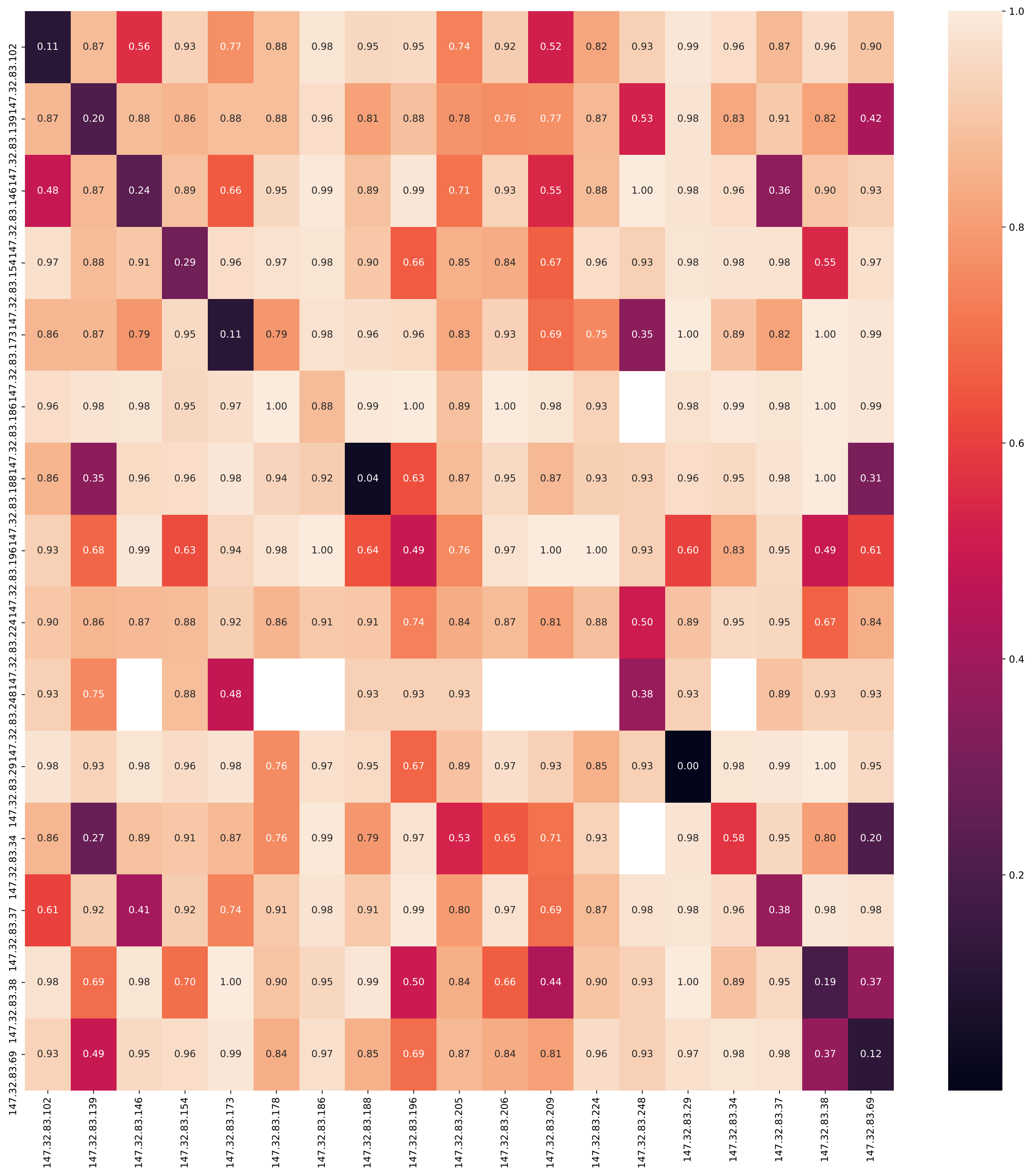


Figure 4.7: Testing experiments comparing the users of dataset C against the users of dataset A. This Figure shows the relationships between users by averaging the profile-to-profile comparison and taking a decision using the threshold. The rows represent IP addresses from the dataset C. The columns represent IP addresses from the dataset A. The white spaces are places where there were no data for comparisons. This is the case, for example, of one user having only TCP flows and a second user only having UDP flows. There are 4 users in dataset A that are not present in dataset C, and this is why the matrix is not a square. The number of profile comparisons used for taking each decision is described in Figure 4.9.

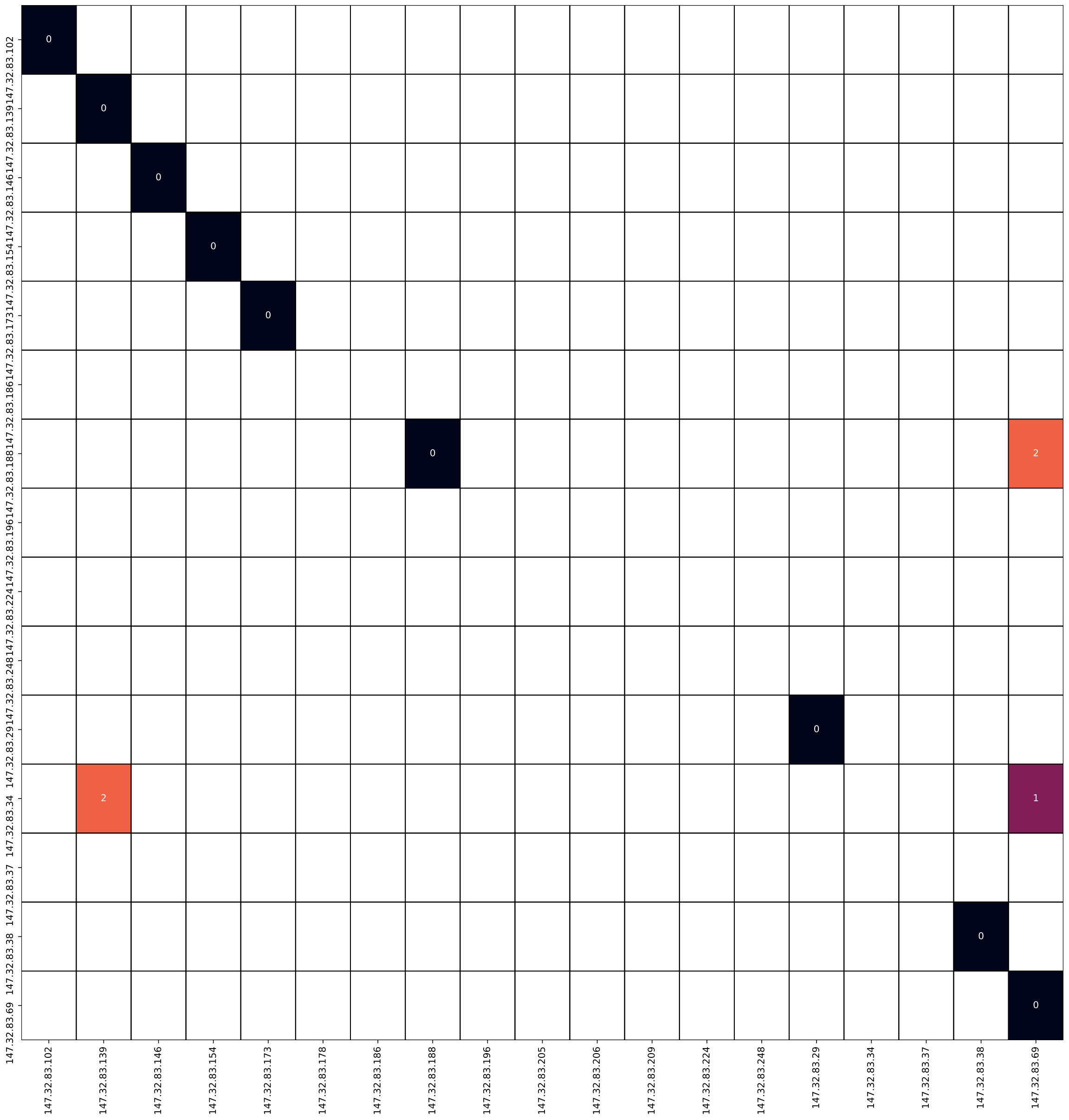
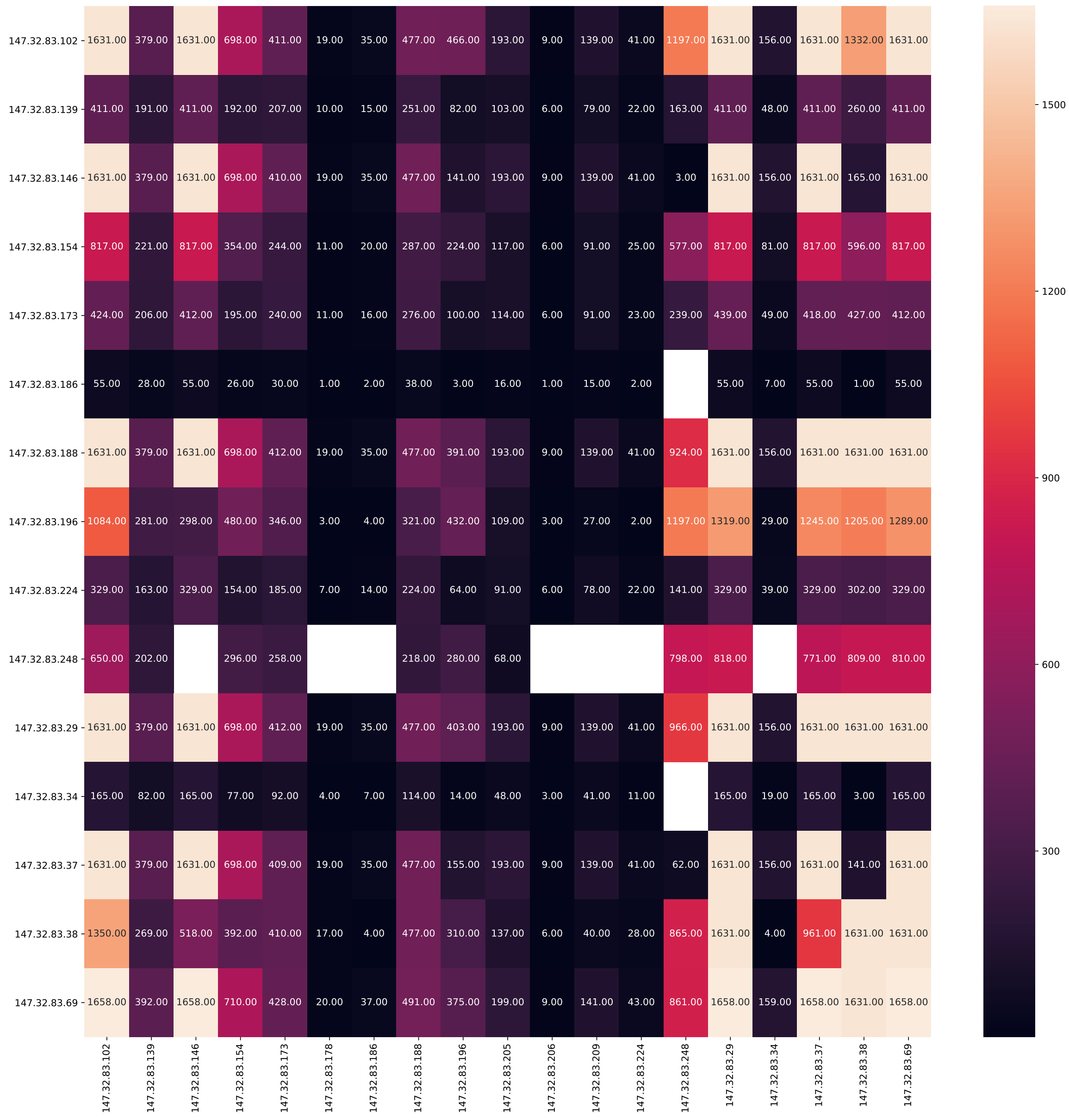


Figure 4.8: This Figure shows the values using the threshold 0.32 on the testing data. The users from dataset C are in the rows of the heat map. The users from the dataset A are in the columns. The results are based on values from the heatmap in Figure 4.7. The 0s show the correctly matched users. The 1s show the wrongly identified users. The 2s show the not minimum values still below the threshold. Being under the threshold means that the IP address 147.32.83.34 is identified as IP address 147.32.83.69 and if we remove the IP address 147.32.83.69 from our known users the IP address 147.32.83.34 would be identified as IP address 147.32.83.139. Two IP address from the dataset C would be matched to the IP address of 147.32.83.69 or respectively of 147.32.83.139. However, both IP addresses 147.32.83.69 and 147.32.83.139 matched themselves more accurately. The IP address 147.32.83.69 is also similar to the IP address 147.32.83.188. Everything else has a higher value than the threshold, or there are no data available.



	147.32.83.102	147.32.83.139	147.32.83.146	147.32.83.154	147.32.83.173	147.32.83.178	147.32.83.186	147.32.83.188	147.32.83.196	147.32.83.205	147.32.83.206	147.32.83.209	147.32.83.224	147.32.83.248	147.32.83.29	147.32.83.34	147.32.83.37	147.32.83.38	147.32.83.69
147.32.83.102	1631.00	379.00	1631.00	698.00	411.00	19.00	35.00	477.00	466.00	193.00	9.00	139.00	41.00	1197.00	1631.00	156.00	1631.00	1332.00	1631.00
147.32.83.139	411.00	191.00	411.00	192.00	207.00	10.00	15.00	251.00	82.00	103.00	6.00	79.00	22.00	163.00	411.00	48.00	411.00	260.00	411.00
147.32.83.146	1631.00	379.00	1631.00	698.00	410.00	19.00	35.00	477.00	141.00	193.00	9.00	139.00	41.00	3.00	1631.00	156.00	1631.00	165.00	1631.00
147.32.83.154	817.00	221.00	817.00	354.00	244.00	11.00	20.00	287.00	224.00	117.00	6.00	91.00	25.00	577.00	817.00	81.00	817.00	596.00	817.00
147.32.83.173	424.00	206.00	412.00	195.00	240.00	11.00	16.00	276.00	100.00	114.00	6.00	91.00	23.00	239.00	439.00	49.00	418.00	427.00	412.00
147.32.83.186	55.00	28.00	55.00	26.00	30.00	1.00	2.00	38.00	3.00	16.00	1.00	15.00	2.00		55.00	7.00	55.00	1.00	55.00
147.32.83.188	1631.00	379.00	1631.00	698.00	412.00	19.00	35.00	477.00	391.00	193.00	9.00	139.00	41.00	924.00	1631.00	156.00	1631.00	1631.00	1631.00
147.32.83.196	1084.00	281.00	298.00	480.00	346.00	3.00	4.00	321.00	432.00	109.00	3.00	27.00	2.00	1197.00	1319.00	29.00	1245.00	1205.00	1289.00
147.32.83.224	329.00	163.00	329.00	154.00	185.00	7.00	14.00	224.00	64.00	91.00	6.00	78.00	22.00	141.00	329.00	39.00	329.00	302.00	329.00
147.32.83.248	650.00	202.00		296.00	258.00			218.00	280.00	68.00				798.00	818.00		771.00	809.00	810.00
147.32.83.29	1631.00	379.00	1631.00	698.00	412.00	19.00	35.00	477.00	403.00	193.00	9.00	139.00	41.00	966.00	1631.00	156.00	1631.00	1631.00	1631.00
147.32.83.34	165.00	82.00	165.00	77.00	92.00	4.00	7.00	114.00	14.00	48.00	3.00	41.00	11.00		165.00	19.00	165.00	3.00	165.00
147.32.83.37	1631.00	379.00	1631.00	698.00	409.00	19.00	35.00	477.00	155.00	193.00	9.00	139.00	41.00	62.00	1631.00	156.00	1631.00	141.00	1631.00
147.32.83.38	1350.00	269.00	518.00	392.00	410.00	17.00	4.00	477.00	310.00	137.00	6.00	40.00	28.00	865.00	1631.00	4.00	961.00	1631.00	1631.00
147.32.83.69	1658.00	392.00	1658.00	710.00	428.00	20.00	37.00	491.00	375.00	199.00	9.00	141.00	43.00	861.00	1658.00	159.00	1658.00	1631.00	1658.00

Figure 4.9: This Figure shows a heat map for all the comparisons done during testing. Each cell shows the sum of all the profile-to-profile probabilities between the corresponding user of dataset C and the corresponding user of dataset A. Every cell has at most 3*28*24 comparisons: 3 is the number of days in dataset C, 28 is the number of days in dataset A, and 24 is the number of hours in the day. In total there can be a maximum of 2016 comparison for two users. This Figure does not show decisions, but amount of comparisons.

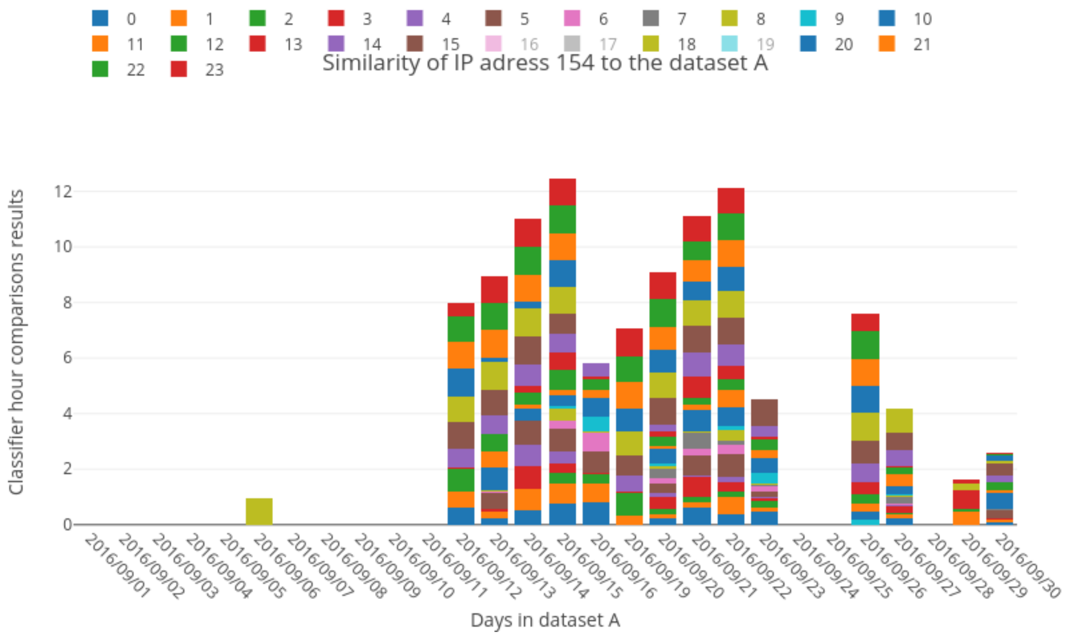


Figure 4.10: Comparison of the IP address 147.32.83.154 from 1st day of dataset B to the all days in the dataset A.

Chapter 5

Conclusion

The identification of users in the network is a long standing problem that has been studied for quite some time. Our thesis focused on analyzing the behavioral features of users in time-windows in order to find similarities in the behaviors. The work started with the capture of real user traffic, which was a very challenging problem. From that traffic we designed the profiles of behavior, we created new features to describe the behaviors and we find a way to compute a distance coefficient between features. This first analysis resulted in a classification method between two profiles. This method was trained and tested and consist in a random forest classifier. The testing results of the classifier achieved 86% accuracy.

With the classification of two profiles ready we went forward and created a method to classify users entirely. For this we designed a method to analyze long term traffic of users and decide if some unknown user can be recognized as part of the known users of our network. This identification problem is very hard due to the nature of human actions. However we were able to achieve a 60% accuracy and a 90% precision.

The success of our algorithms would not be possible without our work labeling each individual user in our dataset by hand and without the visualization tools that we developed. Our tools consist in web applications that can compare and show each individual difference between profiles and users in the network. The tools are complete operative and were published online.

The success of our methods mostly depends on the profile to profile classifier, therefore even small improvements may lead to a better user detection. For this reason we plan to study how to make small adjustments in our technique and find out how they affect the final classification. For example, a replacement of the tree boosting classifier with a neural network may improve our method even further. There is also a possibility of using the k-nearest neighbor (k-NN) with our profiles with a custom distance between the profiles based on the distance between features.

The results show that behavior of users can change during the month. We are not able to identify all of the users because of that change. This can be solved by using different methods than averaging all comparisons. However, it would not help if the behavior of the user is completely different. We may use other features in the future for better identification of users because some users were very similar to each other. Overall the results were good when we take into the consideration that the user identification algorithm was simple.

Bibliography

[1] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos, “Profiling the end host,” *Passive and Active Network Measurement*, pp. 186–196, 2007.

[2] J. Mrkoš, “Computer network client recognition using behavioural analysis,” Master thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2015.

[3] P. Minarik, J. Vykopal, and V. Krmicek, “Improving host profiling with bidirectional flows,” in *Computational Science and Engineering, 2009. CSE’09. International Conference on*, vol. 3. IEEE, 2009, pp. 231–237.

[4] Cisco. (2018) Cisco systems. [Online]. Available: https://www.cisco.com/

[5] ——. (2017) Netflow. [Online]. Available: http://www.cisco.com/go/netflow

[6] C. Bullard. (2017) Argus. [Online]. Available: https://qosient.com/argus/index.shtml

[7] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in *Security and Privacy (SP), 2010 IEEE Symposium on.* IEEE, 2010, pp. 305–316.

[8] R. Anderson, “Why information security is hard-an economic perspective,” in *Computer security applications conference, 2001. acsac 2001. proceedings 17th annual.* IEEE, 2001, pp. 358–365.

[9] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, “Internet traffic classification demystified: myths, caveats, and the best practices,” in *Proceedings of the 2008 ACM CoNEXT conference.* ACM, 2008, p. 11.

[10] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “Blinc: multilevel traffic classification in the dark,” in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 229–240.

[11] Cisco. (2017) Firesight. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/security/firesight/541/user-guide/FireSIGHT-System-UserGuide-v5401.html

[12] D. Kubeša, “parse-binet,” https://github.com/Kobtul/parse-binet/tree/thesis, 2018.

[13] R. B. Netto. (2018) Manati. [Online]. Available: https://www.stratosphereips.org/projects-manati/

[14] V. Paxson, "Bro: a System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999. [Online]. Available: http://www.icir.org/vern/papers/bro-CN99.pdf

[15] D. Kubeša, "Manati," https://github.com/Kobtul/Manati/tree/thesis, 2018.

[16] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L119, pp. 1–88, May 2016. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC

[17] K. G. Derpanis, "The bhattacharyya measure," 2008.

[18] S.-H. Cha, "Taxonomy of nominal type histogram distance measures," *City*, vol. 1, no. 2, p. 1, 2008.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[21] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[22] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[23] A. Jain, "Complete guide to parameter tuning in xgboost (with codes in python)," 2016, [Online; accessed 15-December-2017]. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[24] D. Kubeša, "diploma-profiling-experiments," https://github.com/Kobtul/diploma-profiling-experiments/tree/thesis, 2018.

Appendix A

Contents of attached CD

```
CD
├── README.TXT .................................... File with contents of attached CD
├── text ........................... folder with the text and pdf of the diploma thesis
└── source ............................................... folder with the source code
    ├── parse-binet ............................... parse-binet GitHub repository [12]
    ├── Manati ............................................ Manati GitHub repository [15]
    └── Diploma experiments .......... GitHub repository for running experiments [24]
```