Creating profiles from user network behavior

McDowell, Chad M.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CREATING PROFILES FROM USER NETWORK BEHAVIOR

by

Chad M. McDowell

September 2013

Thesis Advisor:	Robert Beverly
Second Reader:	Geoffrey Xie

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
18–9–2013	Master's Thesis	19-09-2011—27-09-2013

4. TITLE AND SUBTITLE

CREATING PROFILES FROM USER NETWORK BEHAVIOR

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Chad M. McDowell

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.IRB Protocol Number: N/A

14. ABSTRACT

The ability to identify network users based on their network behavior has both positive and negative implications. If users are tracked on the Internet without their knowledge or permission, this could be interpreted as a serious violation of their privacy. If used, however, as part of an organization's network security measures, the ability to identify and verify users might assist in determining whether one user is masquerading as a different user, or whether some user is exhibiting abnormal behavior that might precede malicious insider activity. As a step toward enhancing network security, we investigate the use of DNS hostnames and destination IPs for user identification, based on models of user behavior. Our results indicate that using DNS hostnames is a superior method of modeling user behavior. Additionally, when filtering the data for regular accesses, the accuracies improve for both DNS hostnames and destination IPs.

15. SUBJECT TERMS

Network Behavior Profiles, User Identification

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	127	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

CREATING PROFILES FROM USER NETWORK BEHAVIOR

Chad M. McDowell
Lieutenant, United States Navy
B.S., North Dakota State University, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2013

Author: Chad M. McDowell

Approved by: Robert Beverly
Thesis Advisor

Geoffrey Xie
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to identify network users based on their network behavior has both positive and negative implications. If users are tracked on the Internet without their knowledge or permission, this could be interpreted as a serious violation of their privacy. If used, however, as part of an organization's network security measures, the ability to identify and verify users might assist in determining whether one user is masquerading as a different user, or whether some user is exhibiting abnormal behavior that might precede malicious insider activity. As a step toward enhancing network security, we investigate the use of DNS hostnames and destination IPs for user identification, based on models of user behavior. Our results indicate that using DNS hostnames is a superior method of modeling user behavior. Additionally, when filtering the data for regular accesses, the accuracies improve for both DNS hostnames and destination IPs.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AS Autonomous System
ASN Autonomous System Number
CDN Content Distribution Network
CDF Cumulative Distribution Function
CSU California State University
DNS Domain Name System
IP Internet Protocol
KNN K-Nearest Neigbors
MDI Multiple Daily Instances
NB Naïve Bayes
NPS Naval Postgraduate School
NSA National Security Agency
QNAME Query Name
QTYPE Query Type
RIR Regional Internet Registry
SYN Synchronize
TCP Transmission Control Protocol
tf-idf term frequency-inverse document frequency
UDP User Datagram Protocol
URL Uniform Resource Locator

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

This thesis was made possible by Professor Robert Beverly. I appreciate your guidance and insight, as well as the time you devoted to helping me write the thesis. Thank you for the opportunity to work with you. Thanks also to Professor Geoffrey Xie for providing feedback and new ideas that enhanced the work we were doing.

To my family, thank you for understanding why I seemed to disappear for the last two years. I appreciate your constant support, consideration, and love.

And to my amazing wife, Georgina, I am eternally grateful for your patience and understanding during this time-consuming process. I know the last two years have not been easy for you. Your unwavering support and unquestioning sacrifice are appreciated more than you know, and have served as yet another reminder of how lucky I am to have you in my life.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

The ability to identify network users based on their network behavior has both positive and negative implications. If users are tracked on the Internet without their knowledge or permission, this could be interpreted as a serious violation of their privacy. If used, however, as part of an organization's network security measures, the ability to identify and verify users might assist in determining whether one user is masquerading as a different user, or whether some user is exhibiting abnormal behavior that might precede malicious insider activity.

Identifying patterns of behavior in users is not a new concept. Some online retailers, such as Amazon.com, try to discover patterns in user purchases in order to more effectively advertise other products that the user may be interested in purchasing [1]. In doing so, they ostensibly seek to enhance the user's shopping experience, by allowing the user to discover items that the user wishes to purchase. The benefit to the retailer is the potential for higher sales revenue. User behavior is also tracked in less transparent ways, at least from the point of view of the average user. Companies such as Doubleclick, a Google-owned business, can track users across websites that have signed up to use Google's Adsense service, helping the websites conduct targeted advertising [2].

Tracking users based on behavior might also assist with attempting to mitigate or reduce the insider threat, if conducted as part of a larger security posture for an organization. By assessing the network behavior of users, one might be able to determine if a user's network behavior patterns do not match a historic pattern. The mismatch could imply that a user has logged in with another user's credentials, possibly to avoid attribution for certain network accesses. Alternately, a deviation from past behavior might presage malicious activity by a disgruntled user.

For example, in mid-2013, Edward Snowden revealed sensitive information about projects and activities of the National Security Agency (NSA) [3]. While the full effects of his leaks are still being assessed, his actions have created, if nothing else, a political and diplomatic embarrassment for the United States government. Employed for several years by defense contracting firms that provide services to the NSA, Snowden was able to access vast amounts of top secret information. It is not yet clear exactly when Snowden decided to become a malicious insider,

or whether there was a particular incident that prompted his actions. That said, having worked as a contractor for the NSA for several years, Snowden likely did not start his career with the intention of releasing classified information. If so, he was initially an honest employee and only later, for reasons not yet fully known, decided to release the classified information.

Three years earlier, Bradley Manning, a Private First Class in the United States Army, was arrested for leaking secret military and State Department information to WikiLeaks, a non-profit organization that publishes leaks to its website. He admitted to downloading the information a few months after his transfer to Afghanistan, where he obtained access to the material [4]. As with Snowden, it appears that Manning did not begin his career with the intention of leaking, only deciding to do so later.

1.1 User Network Behavior Profiles

In cases where an insider did not begin employment with the intention of conducting malicious activity, since the insider was acting normally for most of their employment and only began acting maliciously toward the end, it may be possible to generate a profile of normal behavior for an individual user. This profile might allow for the detection of abnormal user behavior, such as a sudden increase in accessing and downloading sensitive files, which could indicate malicious activity.

Research by Banse et al. [5] and Yang [6] indicates that users do exhibit patterns in their network behavior, and that network access profiles can be used to identify users. Their research is discussed in more detail in Chapter 2. Banse et al. focused on DNS queries issued by each user, and Yang worked with destination IP addresses. This thesis draws from aspects of the work by Banse et al. and Yang.

For this thesis, we will be analyzing anonymized packet capture (pcap) files obtained from the Naval Postgraduate School (NPS) network. The goal of our research is to investigate whether focusing solely on user network access data will produce an accurate profile of user activity on our dataset. We also seek to compare and contrast the accuracy of using DNS queries versus destination IP addresses on the same dataset. We will also investigate whether there is an identifiable set of features that can, by themselves, uniquely identify a particular user. Additionally, we will investigate whether these profiles can be used to identify a user who begins to act maliciously.

1.2 Research Questions

This thesis explores whether focusing on network access patterns to create network user profiles can improve the ability to identify abnormal user behavior. In doing so, we investigate the following:

- Will the accuracy of network user profiles be improved by associating a particular source IP with the number of visits to unique destination IPs?
- How does the state of the art perform when applied to NPS network data?
- Are there specific discriminating features that can be shown to identify a particular user?
- Can the user profiles help identify abnormal behavior?

1.3 Significant Findings

The results of our analysis indicate that identification of users based on network traffic is applicable to Naval Postgraduate School (NPS) data, although the accuracy results are lower than when considering residential networks. Our analysis revealed:

- Roughly half of all Transmission Control Protocol (TCP) Synchronizes (SYNs) from a given source Internet Protocol (IP) were only issued during a single network session, and were never issued again during the following seven-day period by that same source. Over half of the Domain Name System (DNS) queries issued by a given source were never queried again by that source in the next seven days.
- Roughly 40% of all TCP SYNs in the dataset were never sent again by *any* source during the seven days following the initial web visit. About 60% of all DNS queries were only issued during a single network session, and were never issued again by *any* source over the next seven days.
- About 10% of SYNs were regularly issued by the same source over the course of seven days. Similarly, about 10% of DNS queries were regularly issued by the same source over the seven-day period.
- Approximately 30% of SYNs were regularly issued by at least one source for at least seven days during the assessment period. About 10% of the DNS queries issued by any source were regularly issued again by at least one source during the next seven days.
- Identification using DNS hostname queries was more accurate than using destination IP addresses for all versions of tests we performed.

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 covers some of the background on the insider threat and discusses related work in the area of creating user profiles based on network behavior.
- Chapter 3 discusses the methodology employed in this thesis, including how our profiles were created.
- Chapter 4 provides the results of our experiments.
- Chapter 5 contains conclusions drawn from the results in Chapter 4 and recommended future work.

CHAPTER 2:
Background and Related Work

This chapter provides an overview of malicious insiders, and current research in the area of user network profiling and identification upon which this thesis is based.

2.1 The Insider Threat

The insider threat is a real and significant challenge facing public and private organizations today [7]. Unlike external hackers, the insider does not need to gain access to the network, and can often access sensitive information by logging on with company-issued credentials, e.g., a username and password. As insiders are authorized users often accessing authorized systems, the activity may not raise suspicion; the users may appear to be performing their normal jobs. Bradley Manning did not conduct any hacking, but merely clicked onto the State Department website on the government's secret network and accessed the documents he chose to leak. While it is not yet known whether Edward Snowden bypassed security controls to access the information he leaked, he was clearly granted access to at least parts of the network. Because both men were granted some form of access, intrusion detection systems would not have alerted, firewalls would have been ineffective, and others would not necessarily think that the men were accessing the information for purposes of leaking it. They could have easily been seen as just doing their jobs.

While Snowden and Manning have grabbed the headlines in the last few years, insiders are clearly not limited to government employees wishing to leak classified information. A 2011 survey by the U.S. Secret Service, CERT Insider Threat Center, CSO Magazine, and Deloitte, found that of cases when the perpetrator of a crime could be identified, 21% of the time it was an insider, and 43% of the respondents reported experiencing an insider incident [8]. The average cost of an insider incident is reportedly $412,000 per incident, and the average victim loses about $15 million per year [9].

2.2 Behavioral Patterns

To mitigate the insider threat, CERT recommends some "best practices," one of which is monitoring and auditing employee actions [10]. One technique is to establish patterns of normal or "good" network behavior by employees. These normal patterns might then be compared against

current behavior, to to see if there is a strong deviation between the two. If so, the user's activity might warrant additional scrutiny to see if the difference is due to benign factors, such as new job tasking, or something more malignant.

These network behavior patterns may not help with insiders who conduct their malicious activities for a prolonged period of time, as their "bad" behavior would be included in their normal pattern. But, studies by CERT have found that more than half of insiders who stole intellectual property did at least some of their malicious behavior within 30 days of termination [11]. CERT has also found that in 80% of cases of IT sabotage, the insider reached the "tipping point" (the point at which they decided to become malicious) within 28 days of his activity [12].

The social aspect of why an employee turns to malicious activity supports the pattern of insiders conducting malicious activity immediately prior to leaving a job. Some known reasons identified by CERT are financial compensation issues, a hostile work environment, problems with supervisor, and being passed over for promotion [13]. As the disgruntled employee wishes to get away from these negative (or perceived negative) factors, it makes sense that he or she would end employment soon, and would conduct the risky malicious activity shortly before leaving.

It may be possible for some tech-savvy insiders to think of ways to circumvent or mitigate such behavioral patterns, perhaps by hiding their behavior or gaining unauthorized access that cannot be traced back to them. But CERT found that most insiders who steal intellectual property (for example) do so using authorized access and are not technically sophisticated [11] [13].

Thus, for many insider attacks, it should be possible to create a historic pattern of good behavior against which their behavior in their final days of employment can be compared. If abnormal behavior is detected in their final days, it might indicate malicious activity. Furthermore, if the behavioral patterns allow for a continual analysis of behavior regardless of termination date, it might be possible to early-detect such insiders as Edward Snowden, whose termination date was known only to him.

2.3 Domain Name System

The DNS is a naming system that translates names that humans can easily remember into IP addresses that are used to locate network devices. This system removes the need for humans to remember numerical IP addresses, allows the domain name to stay the same when IP addresses change, and allows for multiple IP addresses to be associated with one domain name [14]. As a common and simple example, to send a packet to another network device on the Internet, a

user types the domain name into a web browser. If the web browser does not have a cached IP address for that device, the browser sends a DNS request to a DNS resolver to request the IP address associated with the desired domain name. If the resolver does not have a cached IP address for the desired name, the DNS resolver sends a DNS query to a DNS server [15]. If the DNS server has the domain name and IP address in its database, it responds with the IP address.

2.3.1 DNS Query Fields Relevant to Our Analysis

RFC 1035 [15] describes the domain system and protocol. All DNS packets are divided into five sections: Header, Question, Answer, Authority, and Additional. For our analysis, we focused on two of the three fields found in the Question section, QNAME and QTYPE. The QNAME holds a representation of the domain name for which an IP address is being sought, and the QTYPE indicates the type of the query. The types of queries we focused on were A, AAAA, TXT, SRV, and MX records, thinking that these types might reveal unique markers for user network activity. The A records are used to obtain IPv4 addresses, and AAAA records are for IPv6 addresses. A TXT record can be used for human-readable text in a DNS record. Service records (SRV record) imply that the desired record is for a specified server. Finally, MX records are mail exchange records, used for email servers.

2.4 Ways of Gathering Profiles

There are many ways to create user network behavior profiles. For our analysis, we focused on using hostnames in DNS queries and destination IPs found in TCP SYN packets to build patterns of user network behavior. This section describes related work on these and other methods.

2.4.1 Profiles Formed from DNS Queries

Closely related to the work in this thesis are publications by Banse *et al.* [5]. They worked on associating a user with web activity, based solely on websites visited by the user. Taking the point of view of a malicious observer, they collected DNS requests for a full-day to use as their training set. To re-identify those users the following day, they used machine learning software that employed a Multinomial Naïve Bayes classifier along with term-frequency and inverse-document frequency weighting. If the highest probability corresponded to more than one user, Banse *et al.* used the cosine similarity method to select the best fit.

The results of their research indicate that users do exhibit distinguishing behavioral characteristics while using the network. Using a data set averaging over 2,100 daily concurrent users, Banse et al. were able to correctly link 88.2% of user sessions, demonstrating that a curious

DNS resolver would be able to associate a user's current browsing behavior with the same user's behavior on the previous days. Additionally, they showed that the accuracy held up even for long gaps (up to 90 days) between the training set and the test set, suggesting that a user's behavioral pattern changes slowly.

One of the novel approaches taken by Banse *et al.* is that instead of considering the actual destination IP addresses, they looked solely at DNS requests made by each source IP. This is due in part to their desire to mimic an observer with access to a local DNS recursive resolver. This approach offers a key advantage. Many popular web servers have multiple IP addresses, for instance because they are hosted by a content distribution network, a fact that may hide relationships if one is just looking at destination IP address. For example, if a user visits three distinct IPs, but all are for the same online search engine, an analysis of destination IPs may focus too much on the trees versus the forest. By considering only the DNS queries, [16] [5] are able to assess the actual web sites that each user desires to visit.

A potential disadvantage to this technique is that multiple visits to a website may not incur multiple DNS queries, due to DNS caching. Recognizing this, the authors also conducted tests using a simulated cache with DNS records that expire at the end of each day. While this omits multiple daily visits, the knowledge of which might yield value, their model still performed well, with accuracy only dropping from 88.2% to 80.5%. Another downside to the DNS query approach is that destination IPs not obtained via a DNS server, such as a networked printer or other computer on the same network, are not included in the profile. It might be of value to know if a user is suddenly accessing a printer far more often than normal, especially if combined with other anomalous factors. Another possible disadvantage is that this technique requires visibility into the user's DNS traffic. If an analyst is using a network tap that is in the middle of the network, the DNS queries may never be seen. There are several important differences between the work done by Banse *et al.* and this thesis. Banse *et al.* focused on privacy concerns that could result from being able to link anonymous user sessions. While they had access to the identities of the users for validation purposes, their goal was not to demonstrate how to uncover a user's true identity, but rather to be able to associate multiple sessions with a given user, whomever that person may be. This thesis, however, is intended to add to the body of research focusing on malicious insiders. We take the point of view of a network administrator who knows which IP address is assigned to which user, and desires to create a profile for each network user. The ultimate goals are to identify anomalous behavior that could indicate malicious activity, attempts to masquerade as a different user, or simply if Bob is using Alice's computer without

her permission. Perhaps the biggest difference with this thesis and [16] [5] concerns the dataset. Banse *et al.* investigated a residential network for university student housing. Their dataset of DNS requests was populated from 4,153 university students who use the network both for their studies and personal browsing. We hypothesize that a user's browsing pattern at home will yield more uniquely identifiable markers than when that same user is at work. For example, a user might visit a certain news website throughout the day, while at home and at work. However, that same user might not be inclined to read about celebrity gossip or watch streaming music videos while at work, but might regularly do so at home. To that end, a 2008 study by Giroire *et al.* compared user network behavior in distinct environments, such as inside the corporate network, outside the network but using a VPN for access, and being completely outside the network [17]. Their study demonstrated that user behavior indeed varies depending on the user's location.

In contrast to Banse *et al.*, our dataset is from the Computer Science Department at the Naval Postgraduate School. While some students certainly engage in personal browsing during a break, they probably do not visit all the same websites with the same regularity as they might when accessing the Internet from a computer at home. This means that there are likely fewer observable unique markers in our dataset than for a residential network at a civilian university. Thus, we believe our dataset more accurately reflects a corporate or government facility, which is more in line with potential future application to the insider threat.

2.5 Profiles Formed from Destination IPs

Also related to this thesis are publications by Yang [6]. Yang's work was one of the first to focus on building user network profiles from web usage behavior. Using destination IPs visited by users, she created compared the accuracy of three different profiling techniques. Her initial method was to use Weka's learning tree classifier to create user profiles. Yang's other two techniques were to construct profiles using the data mining methods of *support* and *lift*. By using destination IP addresses, Yang is able to capture repeated visits to the same website, which was not possible in the DNS approach discussed in §2.4.1. As with Banse *et al.*, Yang is also concerned about the privacy implications. While she allowed for the possibility of identifying the actual users, the main thrust of her work was directed at privacy concerns for unsuspecting web users.

Yang's work demonstrated that user profiles created with the support and lift methods can be more accurate than learning trees or support vector machines at identifying users. Running tests with variable numbers of concurrent users and training sessions, Yang had three significant

findings. First, for a small number of users, the learning tree method was always most accurate, with 99.30% accuracy for two users with 100 training sessions each. Next, for small numbers of training sessions (but more than two users), the learning tree classifier was most accurate. With one training session, Yang obtained an accuracy of 79.36% for 10 users and 62.90% for 100 users; using 10 training sessions each, the accuracy increased to 90.72% for 10 users and 81.01% for 100 users. Third, as the number of users and training sessions increased, the support-based profiles dominated the accuracy results, with accuracy of 87.36% for 100 users with 100 training sessions.

A major difference between Yang's work and this thesis is with the characteristics of the datasets. Of her initial large dataset, she created sub-datasets that consisted of at most 100 concurrent users [6]. It was also a closed-world environment, where users were only selected if they had at least 300 sessions. Additionally, Yang's impressive accuracy of up to 87% was achieved with 100 training sessions per user. In contrast, our dataset contains a mean of 443 concurrent users for flow traffic, and a mean of 480 concurrent users for DNS traffic. In our tests, we typically use one training session and we do not specifically filter out users with low-levels of activity. Similar to Banse *et al.*, our main training and test sets are composed of one day each, with each day being considered one session, though we did investigate other combinations, such as using training sessions formed from each hour in a day.

Another difference is that the users in her dataset were volunteers who agreed to participate in the study. We believe that when one knows that one's web behavior is being tracked, normal patterns of behavior will be disrupted, at least a bit. If nothing else, some users may elect to self-censor themselves by not visiting certain websites. In the dataset used for this thesis, it is possible that the acceptable-use policy at the Naval Postgraduate School may have encouraged some students to also self-censor, as the military is often quite strict in its enforcement of acceptable online behavior. That said, we believe that the behavior is more likely to be influenced by the commuter status of all students, who can browse as they please after going home.

2.6 Profiles Formed from Information-gathering Software

Some of the related work in this field involves installing information-gathering software on user's computers to record various data, which is subsequently analyzed. For example, McKinney and Reeves [18] used supervised learning techniques to analyze running processes on host computers. Utilizing a Naïve Bayes classifier, they obtained a true positive rate of 96.7% and a false rate of 0.4%.

Similarly, Udoeyop developed user profiles by applying k-means clustering and Kernel Density Estimation algorithms to information collected from host computers [19]. Analyzing data about the host's processor usage, memory usage, hard drive usage, process threads, file system activity, destination IP addresses, and destination port numbers, Udoeyop developed average nominal behavior probabilities for different scenarios in order to identify abnormal behavior.

While these approaches allow analysts to collect a great amount of data, it may not always be practical to install monitoring programs on every authorized computer. Users accessing a corporate or government network remotely from their personal computer may object to installing the software, or they may develop ways to bypass the software. Another consideration with the above research is that the users were volunteers. As indicated in §2.5, we believe that volunteers may consciously or unconsciously alter their behavior. In contrast, this thesis seeks to identify users without the use of information-gathering software.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Methodology

The intent of this thesis was to compare the use of destination IP addresses and DNS queries when attempting to identify users, based solely on network behavior. After collecting user traffic in the form of packet capture (pcap) files, we parsed the traffic to extract the desired data, and performed a tf-idf weighting. We wrote the output to a file formatted for input to the Orange [20] data-mining software. We also performed several analytical variations, such as omitting short-lived accesses, and keeping only long-lived accesses. This chapter describes each of these steps in detail.

3.1 Datasets

The packet captures we obtained were in the form of pcap files collected from the Computer Science building at the Naval Postgraduate School from 12:20 a.m. on May 8 through 12:20 a.m. on May 21. We used C++ with the libpcap library to parse the pcap files for either TCP SYNs using port 80, or User Datagram Protocol (UDP) DNS messages on port 53, extracting the destination IP addresses for the SYNs or hostname queries from the DNS messages, as discussed in §3.2 and §3.3. Table 3.1 shows some descriptive information about the pcaps for flow and dns traffic. The label *dns(mod)* indicates DNS queries that were modified as in §3.4. The DNS pcaps did not included traffic from the local recursive resolver.

3.2 Parsing Pcaps for Destination IPs

For user identification using destination IP addresses, we filtered the port 80 traffic to select only TCP SYN packets, i.e., packets with the SYN flag set to 1, and the ACK flag set to 0. We filtered on SYN-only packets because our analysis was concerned with the number of visits

traffic type	*snaplen (bytes)*	*total bytes*	*total packets*	*unique srcIPs*	*total dstIPs/queries*	*unique dstIPs/queries*
flow	60	408,581,400	5,521,368	640	5,521,368	26,858
dns	1500	576,950,791	5,630,004	611	5,328,788	103,497
dns (mod)	1500	576,950,791	5,630,004	611	5,328,788	50,521

Table 3.1: Dataset statistics for May 8-14. Note that for flow traffic, the number of packets is equal to the total number of dstIPs, as the pcaps were filtered for TCP SYNs. The DNS pcaps were filtered for DNS queries, but only retained A, AAAA, TXT, SRV, and MX records.

Source IP	*Destination IP*	*# of SYNs*
1.1.1.1	2.2.2.2	10
	3.3.3.3	5
	4.4.4.4	2
1.1.2.2	2.2.2.2	8
	5.5.5.5	7
	6.6.6.6	3

Table 3.2: Table representation of the map of maps container for flow traffic.

made to a website. We define "number of visits" to mean the number of TCP flows having the same destination IP address. Had we included the SYN/ACK packets, a single visit to a website with a large amount of data would have yielded more packets than a single visit to a website with a small amount of data, even though the number of visits to both websites was to the same. Additionally, focusing on SYN-only packets approximated Cisco's NetFlow service, and has a lower burden on resources than if we had collected all TCP packets. As client workstations do not typically receive SYN packets, collecting SYN-only packets should yield traffic emanating almost exclusively from user computers. Additionally, by choosing only the SYN packets, vice the subsequent ACK packets, we only log one network access for each TCP session, regardless of how many packets are sent during that individual TCP session. For each TCP SYN packet, we used the C++ map container to track the number of times each source visited each destination IP, by creating a *map of maps*:

```
map <uint32_t, map <uint32_t, uint32_t> > src_dst_map;
```

Table 3.2 shows a graphical representation of how the map of maps container stores the number of SYN packets each source IP made to each destination IP address. The source and destination IPs are notional, as are the number of SYNs.

3.3 Parsing Pcaps for DNS Queries

To use DNS queries for user identification, we selected only packets using the UDP transport protocol and having a source port of 53. The tap for our packet collection was placed between the clients and the local recursive resolver. Thus, collecting UDP on port 53 should yield only DNS queries issued by the clients. For each DNS packet, we manually extracted the object in the Query Name (QNAME) field from the message by skipping over the headers for the

TYPE	Value	Meaning
A	1	(IPv4) host address
MX	15	mail exchange
TXT	16	text strings
SRV	33	server selection
AAAA	28	(IPv6) host address

Table 3.3: TYPE, value, and meaning for selected QTYPE fields

Source IP	*QNAME*	*# of queries*
1.1.1.1	www.siteA.com	10
	siteB.net	5
	siteC.org	2
1.1.2.2	www.siteA.com	8
	siteC.co.uk	7
	siteD.com	3

Table 3.4: Table representation of the map of maps container for DNS queries

Ethernet, IP, UDP, and DNS protocols, and then retrieving the remaining bytes of the packet, which constitute the object being queried. Next, we checked the Query Type (QTYPE) byte field, and if it matched the desired record request, we stored the object. For example, to select only DNS "A" records, we verified that QTYPE = 0x01.

While our initial goal was to focus on queries for IPv4 A records, we accepted queries for AAAA, TXT, SRV, and MX records, to see if the addition of these records would assist Orange in classifying users. Table 3.3 shows the particular QTYPEs and corresponding values [15, 21, 22] of the QTYPE field for the records we selected. As when using destination IPs, we used C++ maps to track the number of times a source IP issued a DNS query for a particular object (see Table 3.4).

```
map <uint32_t, map <string, uint32_t> > src_hostname_map;
```

3.4 DNS Modifications

In addition to selecting various types of record queries, we also recognized that some queries are more descriptive of a user's behavior than others, while other queries can be aggregated together as they represent the same user intent or action. Thus, we explore whether various means of cropping the hostname queries yields more accurate results. 93% of the hostnames were

modified in at least one of the techniques described in this section, and 7% of the hostnames were in a format that was not affected by the cropping.

3.4.1 Retain Second-level Domains

One of the modifications we performed to the DNS QNAMEs was to crop the DNS hostnames to only retain the second-level domains. The goal was to see if the cropped hostnames would more accurately reveal a user's web activity. For example, since `www.google.com` and `google.com` represent queries for the same website, we treat both as just `google.com`. We also observed that a number of DNS queries had numerical prefixes. Some of the numbers appear to be IP addresses, other numbers seemed to be different server names, possibly for load-balancing purposes. Retaining only the parent domain means that queries for, say, `lh4.googleusercontent.com` and `lh5.googleusercontent.com` are treated as just `googleusercontent.com`. By considering similar DNS queries to be the same, we give more weight to the parent domain query, perhaps better reflecting the user's demand to visit that site. We test this hypothesis by comparing results of complete hostnames with cropped hostnames in §4.4 and §4.5.

3.4.2 Exception for nps.edu Websites

An exception to the tested modification of retaining only the parent domain is when the hostname ends with `nps.edu`. As all users on the tested network are students at the Naval Postgraduate School (NPS), there is a very high probability that most users will regularly visit several websites ending with the `nps.edu` domain. Thus, whereas retaining only the parent domain for visits to other websites, including (and, perhaps, especially) those for university websites may help identify particular users, for the NPS network, we retained two more subdomain parts of the hostname.

3.4.3 Remove CDN Domains

If a website is a customer of a Content Distribution Network (CDN), DNS queries for that website may be appended with the CDN's parent domain. For example, an nslookup query for `www.jamestownsun.com`, the website of the local newspaper in Jamestown, North Dakota, shows that the CNAME for this website is `www.jamestownsun.com.edgesuite.net`. The `edgesuite.net` domain belongs to Akamai Technologies, a popular CDN. As the purpose of the DNS request is to obtain an IP address for the content of `jamestownsun.com`, regardless of the involvement of a CDN, we decided to strip off the CDN domain to investigate the effect on user identification accuracy. When combined with the hostname-cropping discussed earlier

in this section, a DNS query for `www.jamestownsun.com.edgesuite.net` is thus recorded as `jamestownsun.com`. The particular CDN domains that repeatedly appeared in the data, and were specifically cropped, were `edgekey.net`, `edgesuite.net`, and `akadns.net`.

3.4.4 Potential Downsides of Using Cropped Hostnames

The cropping techniques discussed above affected about 93% of the hostnames and come with their own potential downsides. For example, queries for `monterey.craigslist.org` and `fargo.craigslist.org` both reveal a user's interest in craigslist. By retaining only the second-level domain, we effectively increase the weight of `craigslist.org`, but we lose the potential unique marker of which city or region the user really wants to view. Additionally, each craigslist page has links associated with a particular region, such as nearby cities, and each link entails a DNS query when the page is loaded. By considering only the second-level domain of each DNS query, this method could potentially overweight `craigslist.org`, when the DNS queries for each link on each page are logged. Thus, a user who navigates to a particular city on the website will have logged dozens of hits (in our model) to `craigslist.org` for a single visit. That same user could conceivably make multiple visits to `cnn.com`, but the craigslist marker might inappropriately dominate the user's profile.

Another potential pitfall is that the modifications might be overly specific to our current dataset. Selected DNS queries over a week's time are not enough to develop a sound model of the all the data in the dataset, or of potential future data. Finally, if DNS queries are sent to different servers for load-balancing purposes, it is not clear what the longevity of a single server would be. If queries for the different servers change often, then reducing them to the same second-level domain might reap benefits. On the other hand, if there are more DNS queries for one particular server over the others, then by stripping the query of the server name could mean the potential loss of a unique marker with no benefit to identification accuracy.

3.5 Tab Files

The Orange data-mining software program reads input from *tab* files [20]. The tab files are simple text files that are formatted in the proper manner, and have ".tab" as their extension. Table 3.5 shows an example tab file. For our analyses, we created tab files that were organized in a table-style format, with a header row consisting of each unique destination IP visited by all users during the training session. Following the method of Banse *et al.* for most of our training and testing sets, this period was one 24-hour day. The final column of row one merely holds the

SiteA	*SiteB*	*SiteC*	Src
continuous	continuous	continuous	discrete
0	7	9	*srcIP1*
0	10	0	*srcIP2*

Table 3.5: Example format of an Orange tab file

title of the the value to be tested. In our case, it says "src," indicating that we wish to identify source IP addresses.

The second row of the header indicates whether each datum in the first row has few or many possible values, indicated with the label "continuous" or "discrete." For instance, the row one header cell containing "src" is labeled as discrete, since there are a discrete number of source IP addresses in each session. The remaining row one cells, all of which contain destination IP addresses, are considered to be continuous, since the number of visits to an IP address is virtually unrestrained.

After the header, the first (or, optionally, the last) column of the tab file is populated with the object to be tested, which is the source IP addresses for us. Each cell under the destination IPs in the header is filled in with the tf-idf weighting for the number of times the source IP in column one visited the destination IP in the header.

3.6 Term Frequency-Inverse Document Frequency

tf-idf is a measure of how distinctive a term is in a document. In our case, it is a measure of how distinctive a particular destination IP or DNS query is to a user's network profile. There are several variations on this metric [23]. As we were attempting to reproduce (and further compare) the user identification technique set forth in Banse *et al.* [5], we employed the method they chose, and we describe their method next. Term frequency *tf* is a measure of the frequency f of occurrences of a term t in a given document d. The more frequently a destination IP or DNS query (the term) is accessed during a user's on-line session (the document), the more important that IP is to a user's profile. The frequency $f_{t,d}$ is a raw count of the number of times t occurs in d, and thus may be biased toward longer documents that have more absolute occurrences of t, due to length. To help mitigate this bias, we employ logarithmic scaling to create $tf_{t,d} = log(1 + f_{t,d})$ [24].

The inverse document frequency *idf* is a measure of the frequency of occurences of term t across all documents D. A destination IP or DNS query hostname that is common in all user sessions

SiteA	*SiteB*	*SiteC*	Src
continuous	continuous	continuous	discrete
0	1.55	3.23	*srcIP1*
0	2.01	0	*srcIP2*

Table 3.6: Example format of an Orange tab file with tf-idf applied

(such as the school's website or webmail site) is less valuable for individual user identification than an IP that is common to a small number of users. The *idf* for a particular term is computed by dividing the total number of documents N by the number of documents containing the term, $d \in D : t \in d$, and then taking the logarithm of the quotient: $idf = log(N/(d \in D : t \in d))$. Taking term frequency and inverse document frequency together [24], the tf-idf is:

$$\text{tf-idf} = log(1 + f_{t,d}) \cdot \frac{N}{d \in D : t \in d}$$

Use of the tf-idf weighting should increase the effect of a destination IP address or DNS query hostname that occurs frequently in a user's on-line session, while simultaneously reducing the effect of a destination IP or DNS hostname that is commonly accessed or queried by many different users. Table 3.6 shows a notional example of how the tab file looks after the tf-idf weighting is applied.

We performed the tf-idf transformation prior to creating and writing to the tab files. The C++ maps referenced in §3.2 and §3.3 already contain the number of visits for each source IP to each destination IP (or queries for each hostname). As we were parsing the pcap files for each session, we used another map to keep track of the number of source IPs that visited each destination IP. The total number of source IPs can be obtained just from the size of the map. We used these frequency counts to perform the transformation as we were writing to the tab file.

3.7 Training and Testing Time Periods

This section describes the different time-periods we used for the training sets and tests sets. Most of our analyses were performed by training on one day, and testing on the next day. Each day was considered one session for each user. We also performed variations on the training sets, such as increasing the number of recognized sessions per day, and using multiple days for the training set.

3.7.1 Daily Instances

In their work, Banse *et al.* were taking the point of view of a malicious or "curious" DNS resolver. For their training and testing datasets, they used a time period, or *epochs*, of one day for the training set, and the immediately following day for the test set [5]. Following their example, our primary epochs for training and testing sets are two adjacent days. In both train and test datasets, we create an array, or *instance*, for each source IP, consisting of every destination IP (or DNS query) that source visited (or issued) throughout the 24-hour epoch. In these tab files with daily instances, each active source IP address only appears one time. While the length of the epoch is 24 hours, the epoch itself need not begin at midnight. That said, we elected to start our epochs at midnight in our local time zone.

3.7.2 Multiple Daily Instances

We also explored the effect of setting the epoch to 24 hours, but having the training set cover multiple days to determine whether more training samples would yield higher accuracies. The effect is similar to merging adjacent tab files, provided we updated the tab file header to reflect all destination IPs that were seen throughout the entire training period.

We created tab files with multiple daily instances for training sets only, and kept the test sets at one 24-hour day. The number of days covered by the training set ranged from two to six successive days. That is, if the first day was the i-th day, d_i, then all the days in the training set are $d_i, d_{i+1}, d_{i+2}, ..., d_{i+j}$, where j is the total number of days being considered in the training set. The test set would then consist of day d_{i+j+1}, the day immediately following the last day in the training set.

3.7.3 Advantages and Disadvantages of the Different Methods

There are several advantages to using the daily instances described in §3.7.1. Perhaps the largest advantage is that a daily instance incorporates all network activity for a user over a full day, and uses that model to identify users on the following day. Since the training set is a full day, we obtain a model of a user's daily activity, regardless of time of day.

The model using multiple daily instances has the potential to produce a profile of normal daily use. Banse *et al.* found that their profiles performed well, even after a 90-day gap between training day and testing day [5]. However, their dataset was obtained from a residential housing network at a university. In contrast, our dataset is from the Computer Science building at a

commuter school. The amount of private browsing is almost certainly less than in a residential network, as discussed in §2.5.

3.8 Classifiers

Identifying users based on network behavior can be treated as a multi-class classification problem. We calculated accuracy using three different classifiers: Naïve Bayes (NB), Decision Tree, and K-Nearest Neighbors (KNN). These classifiers were selected due to their use by the papers from which this thesis drew some inspiration. Banse *et al.* [5] used Naïve Bayes for their user identification accuracies using DNS queries. Yang [6] employed the learning tree classifier (in addition to profiles using support and lift) in her method using destination IP addresses. K-nearest neighbor was included due to its use by Udoeyop [19]. Despite employing significantly different methods than this thesis, as discussed in §2.6, Udoeyop's use of KNN made it an attractive third test.

3.8.1 Naïve Bayes

When using Naïve Bayes to identify users, we consider all the source IPs and destination IPs (or DNS queries) in a specified time epoch. The source IPs are treated as the classes, and the destiation IPs are the features in a feature vector. The Naïve Bayes formula yields the class with the highest probability, based on the probability of the observed features and the relative probability of each class [25]:

$$\arg\max_{C} \left[P(C \mid \vec{F}) \right] = \arg\max_{C} \left[P(C) \prod_{i=1}^{n} P(F_i \mid C) \right]$$

where the class C is a source IP address, and feature vector $\vec{F}$ is a vector holding destination IPs or DNS query hostnames. For our purposes, the Naïve Bayes classifier should yield the most likely user source IP, given the probabilities of the observed destination IPs or hostname for each user.

3.8.2 Classification Tree

A decision tree classifies instances based on the features in a feature vector, sorting the classes based on the features [26]. As in Naïve Bayes, the features are the destination IPs or DNS queries, and the class is the source IP of the user. At its basic level, the classification tree will look at all the destination IPs in the training set that were visited by each source IP. It will use

that information to come up with a series of "if then" statements that it will use on the testing dataset. Orange uses the C4.5 tree induction algorithm in its classification trees [27].

3.8.3 K-Nearest Neighbors

With the K-nearest neighbor (KNN) method, the algorithm finds k neighbors that are nearest to the instance in question. If $k > 1$, then a plurality vote is taken of these k nearest neighbors [25]. While KNN is a relatively simple classifier, all of our features are of the same type. That is, we are only using destination IPs or DNS queried hostnames for our features. The hope is that this will avoid some of the issues with using KNN to find nearest neighbors, when some features are have significantly different implications. For example, if we were included features such as the user's age and salary, in addition to the destination IPs, the KNN model could potentially consider two sources to be similar, based mainly on age and salary, but not destination IPs [28].

3.9 Orange

For our accuracy calculations, we used Orange [20], an open-source data-mining software program. Although Orange has the option of using a graphical user interface (GUI) for visual programming, we utilized Python scripting to run our tests, as it gave us a greater degree of control, and allowed for us to create reports in a tailored format for subsequent processing.

3.9.1 Settings for NB, Tree, and KNN

The particular settings we used in Orange for Naïve Bayes, Classification Tree, and K-Nearest Neighbors are as follows:

```
bayes = Orange.classification.bayes.NaiveLearner(train_data, m=2)
tree450 = orngTree.TreeLearner(train_data, max_depth=450,
same_majority_pruning=1, m_pruning=2)
knn5 = Orange.classification.knn.kNNLearner(train_data, k=5)
```

The settings used for our classifiers were obtained by running trials using different settings, and choosing the ones that yield the highest accuracies. We varied the *m-estimate* in Naïve Bayes, *m for pruning* in Learning Tree, and how many k nearest neighbors to select from in KNN, before settling on the above values. Of note, the settings for Learning Tree did not seem to affect the accuracy when training on one day and testing on the next, but did yield significant accuracy increases when the training set included multiple daily instances. Also, variations in the *m-estimate* for Naïve Bayes appeared to have no significant effect on any of our tests.

3.10 Lifetime

For each source and destination IP pair, we were interested in knowing whether a particular source ever visited that same destination IP again. If so, we wanted to determine the length of time between the first and last time that source visited that destination IP. We also considered visits to each destination IP, regardless of the particular source. In other word, we wanted to know whether there are certain destination IPs that a user only visits often, or only a few times, and whether there are certain destination IPs that are never visited more than once by any user.

3.10.1 Definition of Lifetime Using Sliding Windows

For a particular source/destination IP pair, we define the *lifetime* to be the amount of time that passed between the first and last time that destination IP is seen in a TCP SYN packet sent by that source IP. The time period is a one-week *sliding window*. For example, if the assessment period begins on Wednesday, May 8, we look at all destination IPs seen between May 8-14 (Wednesday through Tuesday), noting also the source IP in the packet. We next look at all source/destination IPs seen between May 9-15 (Thursday through Wednesday). We continue until the starting date has ranged an entire week (see Figure 3.1).

We do a similar calculation for each destination IP, regardless of source, using the same sliding window technique. We also calculate lifetimes for source IPs and the hostnames seen in DNS queries issued by those sources (source/hostname pairs), as well as for DNS query hostnames, regardless of source. Full lists were created of lifetimes for src/dst IP pairs, src/hostname pairs, dst IP regardless of source, and hostname regardless of source.

3.10.2 Recording Lifetimes Using Sliding Windows

Once a destination IP is first seen in a window, it is recorded only for that window, and is not considered in any other window. This is done to ensure we calculate only one lifetime per IP. For instance, as shown in Table 3.7, if destination IP *A* is seen on Wednesday, Thursday, Friday, Saturday, and Sunday (all at the same time of day), the lifetime for *A* would be four days. If destination IP *B* is seen on Thursday, Friday, and Saturday (at the same time of day), it has a lifetime of three days. Although *A* was seen on Thursday, which was the starting day for a new sliding window (the one that *B* is in), *A* is ignored for *B*'s window. If this was not the case, we would register multiple lifetimes for *A*, each in different windows.

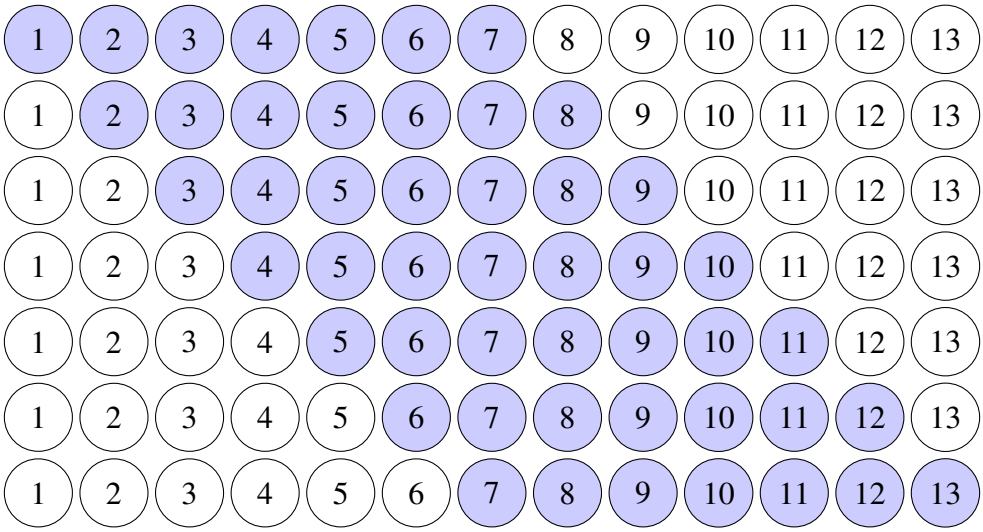

Figure 3.1: Graphic showing the sliding windows. Each node represents one day, with shaded nodes indicating days included in a particular seven-day window. Each window covers seven days, with start days on each day of a week (the first through the seventh of this month).

dstIP	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Lifetime
A	x	x	x	x	x			4 days
B		x	x	x				3 days
C	x						x	6 days

Table 3.7: Destination IPs *A* and *C* are recorded in the Wednesday-Tuesday sliding window; dstIP *B* is recorded in the Thursday-Wednesday sliding window. Dst IP *A* is not considered in the Thursday-Wednesday window.

3.10.3 Method of Calculation for Lifetimes Using Sliding Windows

To calculate the lifetime, we used a C++ map. The key was the pair <*srcIP, dstIP*> and the value was the pair <*time first seen, time last seen*>. The first time a src/dst IP pair is seen, *time first seen* and *time last seen* are set to the same time. For subsequent occurrences of a src/dst pair, only the *time last seen* field is updated. We use a separate map for each window, for a total of seven maps (one for each day of the week). After creating maps starting on each day of the week, we calculate the lifetime by taking the difference *time last seen* - *time first seen*. If a source only sent one SYN to a certain destination IP, then the lifetime for that destination would be zero, since *time last seen* and *time first seen* would be equal.

3.10.4 Standard Definition of Lifetime

In addition to tracking lifetimes using a one-week sliding window, we also calculated lifetimes purely within the context of a fixed one-week period. In this method, all destination IPs were included in the same tracking map, regardless of the day on which they were first seen. For example, for the week May 8-14, destination IPs seen on May 10 were tracked in the same map as those seen on May 8, and the last possible date to search for any IP was May 14.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Results

The goal of this thesis was to explore the differences between using destination IPs and DNS hostnames when attempting to identify network users. In doing so, we investigated the effect on accuracy when filtering the data sets to omit infrequently seen destination IPs and DNS queries, and when filtering to retain only destination IPs and DNS queries that were seen regularly during the observed period.

In the results below, we see that the best improvements in accuracy were when the filters either included or removed source/destination IP pairs or source/DNS query hostname pairs, versus just looking at either the destination IP or hostname. Additionally, the accuracy improved when the datasets were filtered for more frequently observed src/dst IP pairs or src/hostname pairs, but the tradeoff was that fewer source IPs were present in those datasets.

4.1 CDFs for Access Lifetimes

We generated Cumulative Distribution Functions (CDFs) for the lifetimes for src/dst IP pairs, src/hostname pairs, dst IPs irrespective of source, and hostnames irrespective of source (discussed in §3.10). For convenience, we will refer to src/dst IP pairs as *src/dst*, src/hostname pairs as *src/host*, and dst IPs (or hostnames) irrespective of source, as *dst-only* (or *hostname-only*).

4.1.1 CDFs for Lifetimes of Flow Accesses

Figure 4.1 plots the CDFs for src/dst IP pairs and dst-only accesses, for both the standard and sliding-window lifetimes. As discussed in §3.10, we defined the standard lifetime to be the time difference between the *last seen time* and *first seen time* for a destination IP over the course of one week. As the value of *last seen time* is updated each time the dst IP is observed during the week, we are effectively using *max(last seen time - first seen time), 1 week)*. The sliding-window lifetime is also equal to *max(last seen time - first seen time), 1 week*, but is calculated by looking seven days out from each day in the one-week assessment period.

When the source IP is considered for the sliding-window lifetimes, about 50% of the <src/dst> pairs had very short lifetimes, less than 371 seconds (about six minutes), and about 10% had lifetimes longer than six days. For the standard lifetimes, the bottom 55% had lifetimes less

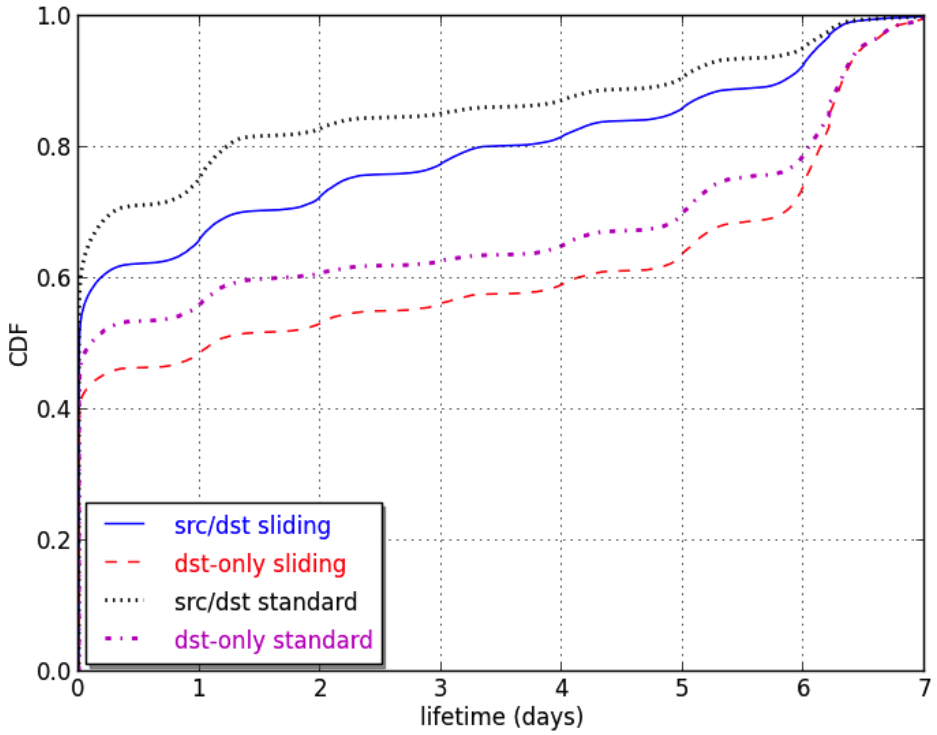

Figure 4.1: Flow - CDF of standard and sliding-window lifetimes for src/dst and dst-only accesses (5-14 May)

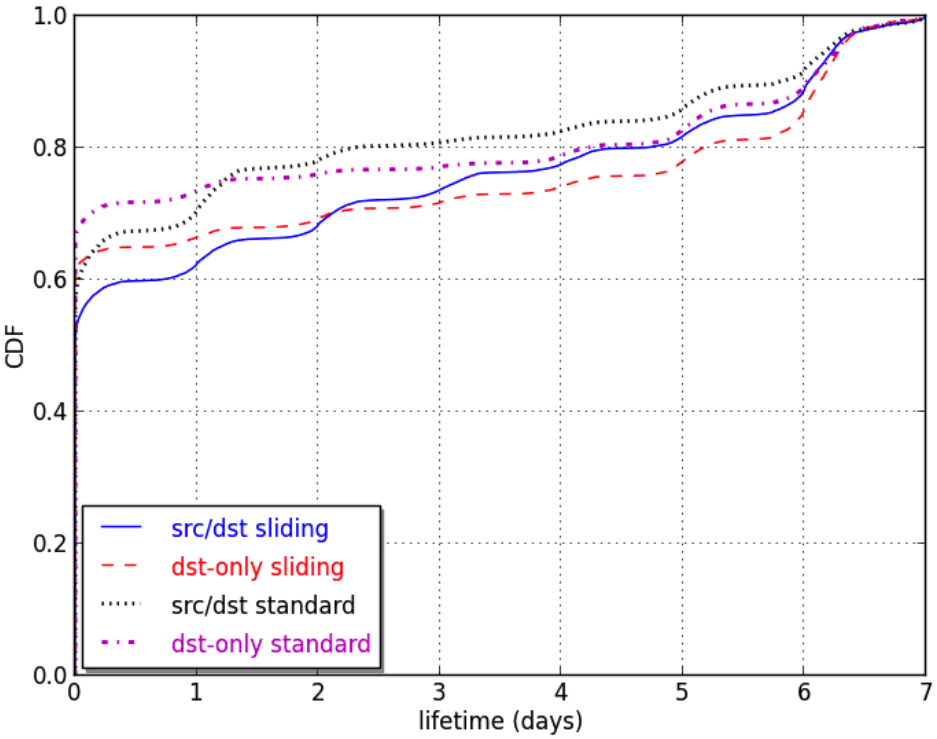

Figure 4.2: DNS - CDF of standard and sliding-window lifetimes for src/hostname and hostname-only DNS queries (5-14 May)

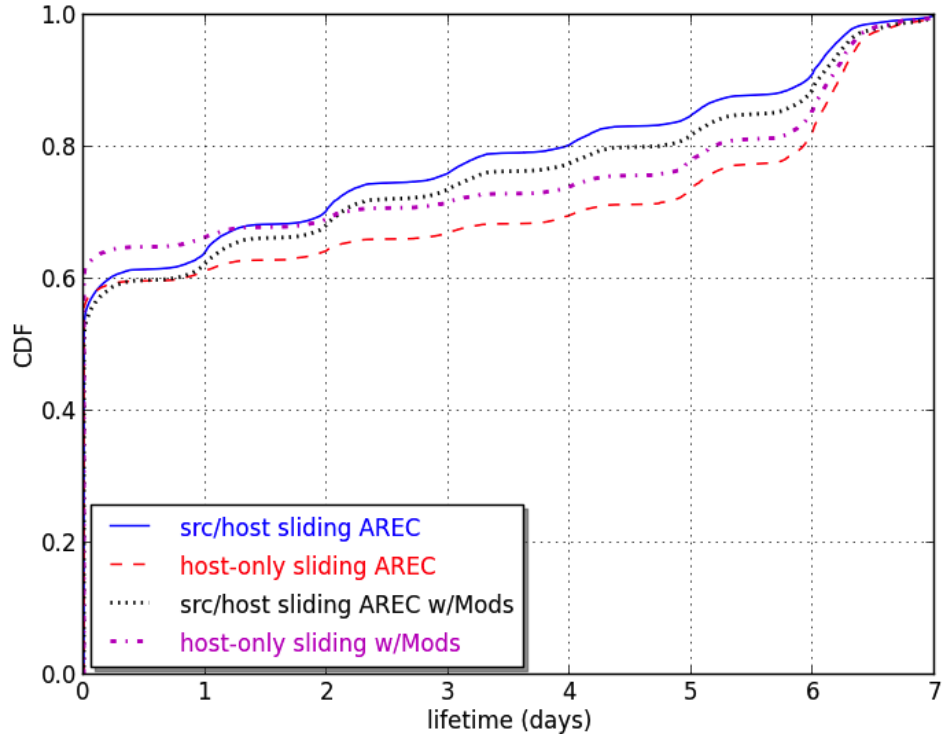

Figure 4.3: DNS - CDFs of sliding-window lifetimes for complete and modified DNS queries (5-14 May)

than 257 seconds (see Table 4.1), and about 5% were longer than six days. The graphs also displayed a stair-step appearance. For Saturday and Sunday (the third and fourth days), the steps are still present, albeit less pronounced for the standard lifetimes.

It makes sense that the sliding-window lifetimes for src/dst pairs display a more consistent stairstep shape than when using standard lifetimes. Suppose a src/dst pair *A* was first active on the sixth day of the week being analyzed (our *base week*), and that *A* was active on each subsequent day. Using standard lifetimes, since observations are stopped on the last day of the one-week period, *A* can only achieve a max lifetime of one day, even though it was active for a week after first appearing in the base week. Under sliding-window lifetimes, since we look one week out from each day in the base week, *A* is able to obtain a max lifetime of seven days. It also makes sense that the CDF for src/dst pairs has noticeably higher y-values than the CDF for dst-only. The src/dst pairs are more specific, so we had expected shorter lifetimes, which is reflected in the higher y-values for the src/dst CDF.

Data	*Mode*	*Bottom Pct*	*Sec*	*Min*
flow	src/dst	50%	371	6.2
	dst-only	40%	653	10.9
dns	src/host	60%	1,446	24.1
	host-only	57%	97	1.6
dns (mod)	src/host	50%	88	1.4
	host-only	60%	62	1.0

Table 4.1: Sliding-window lifetimes for src/dst, src/hostname, dst-only, and hostname-only, showing the time corresponding to the point at which the CDF appears to break away from the y-axis.

For destination IP lifetimes created irrespective of the source IP, the CDFs have a similar appearance. The sliding-window lifetime CDF shows that roughly 40% of all destinations had very short lifetimes, less than 653 seconds, with about 30% more than seven days. For the standard lifetime, the numbers were about 50% below 577 seconds, and 20% over seven days. The dst-only graphs have a similar stair-step look, but with more dramatic jumps at the end. As with the src/dst pairs, the sliding-window lifetimes for dst-only are more regular than the standard lifetimes, though the steps are less pronounced than for src/dst pairs.

4.1.2 CDFs for Lifetimes of DNS Queries

Figure 4.2 shows the standard and sliding-window lifetimes of DNS queries using src/hostname pairs and hostname-only. The shapes are similar to those using destination IPs, except that the differences among the DNS query graphs are slight, regardless of the type of lifetime or whether source IPs are included in the assessment.

Notice also that, unlike CDFs for destination IP the host-only CDF starts out higher than the src/host CDF, then crosses below. A possible explanation is DNS Prefetching and TCP Preconnecting (see discussion in §4.1.3). A website could contain many Uniform Resource Locator (URL) links, all of which could cause the browser to issue DNS queries. If those URLs are mostly subdomains for the website currently being viewed, many IPs for those subdomains could be the same. If so, a single visit to a webpage could cause significantly more DNS queries than TCP SYNs. For example, we did a test using Google Chrome to access `monterey.craigslist.com`, and observed 19 different DNS queries, but only one SYN. In the graphs using destination IPs, the CDFs had greater separation from one another. Figure 4.3 shows a comparison between DNS queries for A records, and DNS queries when the hostname has been modified, as discussed in §3.4. It can be seen that the modified DNS hostnames have

longer lifetimes for the src/host pairs, as indicated by the CDF having smaller y-values. A likely explanation is that cropping the hostnames combined some different queries into a single query. The *last seen time* and *first seen time* for any of those queries was then given to the cropped hostname, i.e., the cropped hostname had more opportunities to record a longer lifetime. one As with the CDFs for flow data, we notice that about 50-60% of the hostnames were only queried in short user sessions, with lifetimes of only about 65-82 seconds when the source IP is included, and about 35-49 seconds for the hostname-only CDFs.

4.1.3 Implications of the Sliding-window Lifetime CDFs

There are several implications that can be drawn from the shape of the graphs. First, and perhaps most noticeably, is that half of the destination IPs that were accessed by a particular source IP were only accessed during short network sessions. This implies that when the average user visited a website or accessed a destination IP, about half the time that same user never accessed the dst IP again during the next seven days. As discussed below, TCP Preconnecting, embedded URLs and advertisements may all trigger multiple TCP SYNs. Furthermore, when any user accessed a destination IP, there was roughly a 40% chance that that destination IP would not be accessed again by *any* user. Much of this could be due to CDNs and websites that have multiple load-sharing IP addresses, reducing the likelyhood that the same (or any) user would access those same dst IPs. When using DNS queries for the analysis, it is also seen that half of the queries are never repeated by the same source IP again in the next seven days, and 60% of all queries are never repeated by any source IP.

A possible explanation for the large number of non-repeated accesses is DNS Prefetching and TCP Preconnecting [29]. Many webpages have links to other webpages. In an effort to speed up users' browsing experience, some web browsers will issue DNS requests for the linked pages, and potentially establish a TCP connection, even if the user does not actually click on the links. Thus, if a user navigates to some webpage and never returns, potentially dozens of TCP SYNs and DNS queries are sent, all due to the user visiting that single page. If the user never returns to that website, then those same SYNs and DNS queries may never be repeated. Additionally, a webpage may contain embedded URLs or advertisements that automatically send DNS requests and TCP SYNs to deliver the content. Again, if a user does not return to the webpage, those DNS requests and TCP SYNs might not be repeated. Furthermore, the webpages or advertisements in the webpage themselves may use CDNs, which could lead to a greater variety of destination IPs for the TCP SYNs.

Second, the presence of the stair-step shape indicates that most of the repeat dst IP or hostname accesses occurred 18-30 hours after the first access, which make sense. The dataset is from an academic building, and relatively few students or staff are in the building late at night. Thus, the 18-30 hour repeat is a valid expectation for initial accesses that occur during the workday. This same stair step was also present with the DNS queries, which again makes sense. Each step involves roughly 5% of the accesses or DNS queries. Third, about 10% of the accesses involve the same source IP visiting the same destination IP at least twice in a period of at least six days, and about 30% of all destination IPs (irrrespective of source IP) in the dataset were accessed at least twice during a six-day period.

4.2 Ignoring or Retaining Specific Accesses

Using the list of lifetimes for flow and DNS access data (§3.10), we can elect to filter out the short-lived src/dst pairs, src/hostname pairs, and short-lived destination IPs and hostnames (irrespective of source IP). As these short-lived IPs or queries were not repeated after the initial access, their presence likely detracts from the robustness of our generated models. For example, if a source visits a specific destination only one time, then that src/dst pair will be present either in a training instance or a testing instance, but not both. Thus, omitting these short-lived accesses could help with identification accuracy.

We can also filter the datasets to only retain longer-lived activity. Because these accesses were more frequently accessed, they indicate more durable user network behavior. Creating training and testing datasets composed only of the long-lived accesses should facilitate source identification accuracy. That said, because our lifetime definition is *last seen time - first seen time*, it is possible for a destination IP or DNS query to have been only issued twice: once on the first day, and once on the last day. These IPs or queries would then be included alongside accesses that occurred every day. Despite this concern, filtering for long-lived accesses should at least remove the destinations or queries that are clearly less durable, which should have a positive effect. In §4.4.4, we examine the accuracy results when considering the number of different days in a week each dst IP or DNS query was observed.

Since our analysis uses packet captures, and is not occurring in real time, we are able to create the lists of short/long-lived accesses after the the fact, and then optionally apply the filters to retain or omit sources or destinations (or hostnames) from one or both of the training and test sets. For brevity, we will use the abbreviation *SL* for short-lived and *LL* for long-lived, when the meaning is clear from the context.

Note that we are considering the short-lived accesses to be those 50-60% of accesses with extremely short lifetimes (Table 4.1 that caused the CDFs in Figures 4.1, 4.2, and 4.3 to appear to initially cling to the y-axis. Long-lived accesses are those that have lifetimes of at least five or six days. There are, of course, accesses with lifetimes that we did not classify as either short-lived or long-lived. When the SL accesses are omitted, the dataset then contains the LL accesses as well as the "in-between" accesses. When just the LL accesses are retained, both the SL and in-between acceesses are omitted from the dataset. The variations we explored for the flow data were as follows:

- filter just the training sets to omit short-lived src/dst pairs
- filter both training and testing sets to omit short-lived src/dst pairs
- filter just the training sets to omit short-lived dst IPs only
- filter both training and testing sets to omit short-lived dst IPs
- filter just the training sets to retain long-lived src/dst pairs
- filter both training and testing sets to retain long-lived src/dst pairs
- filter just the training sets to retain long-lived dst IPs only
- filter both training and testing sets to retain long-lived dst IPs only

For the DNS query data, we performed the same variations as with the flow data, but using src/hostname pairs and hostnames only, vice destination IPs.

4.3 Unassisted Accuracy

The first experiment we performed was written in C++, and was "unassisted," in that it did not utilize any data-mining software packages. It was performed only on destination IP addresses, and did not use access lifetimes to omit or retain destination IPs. We performed two analyses on our datasets. The first analysis did not involve any filtering of the training or test sets. For the second analysis, the test sets were filtered to only retain source and destination IPs present in the training set.

Each training set was composed of TCP/IP source/destination pairs covering a full day of TCP SYN packets. The test sets were built from src/dst pairs from the immediately following day, and were constructed in the same manner as the training set, except that for the filtered analaysis, the source and destination IPs are only added to the test set if they also exist in the training set. Note that this method was only employed for this particular test, as it creates an unfair advantage to accuracy, as new source IPs would never be classied. The goal was to establish a

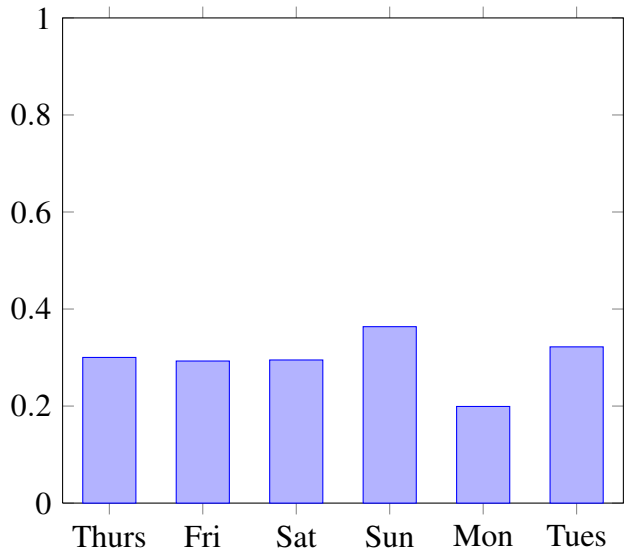

Figure 4.4: Day-to-day accuracy using tf-idf and cosine similarity on src/dst IPs with unfiltered datasets (May 8-14).

baseline with a high, albeit unfairly obtained, identification accuracy using destination IPs, and determine how the data-mining software package Orange would fare against this method. The program creates an array of destination IPs visited by a particular source IP. In classifying the test set, we make a prediction for each source, by applying tf-idf and cosine similarity to the accompanying array of destination IPs. The tf-idf was calculated as described in §3.6.

Results are shown in Figure 4.5 for May 8-14, 2013. When both the training and test sets were from a weekday, the accuracies were fairly consistent, averaging 30.5% for the unfiltered analysis and 34.6% for the filtered version. Accuracy decreased when the training set was on Friday and the testing set was on Saturday, rebounded when training/testing on Saturday/Sunday, then decreased for training/testing on Sunday/Monday, before stabilizing for training/testing on Monday/Tuesday. The mean average for the full six-day period was 29.5% for the unfiltered analysis and 33.5% for the filtered version. While these accuracy results are not bad, we were surprised that the filtered analysis was only 4% higher than the unfiltered tests. Our intuition was that by removing sources and destinations in the test sets that were not present in the training sets, we lowered the chances of wrongly classifying a source.

The noticeable decreases in accuracy for training/testing on Friday/Saturday and Sunday/Monday can reasonably be explained by the dataset. The packet captures that were analyzed were from an academic building which sees few students during the weekend. Thus, the training set

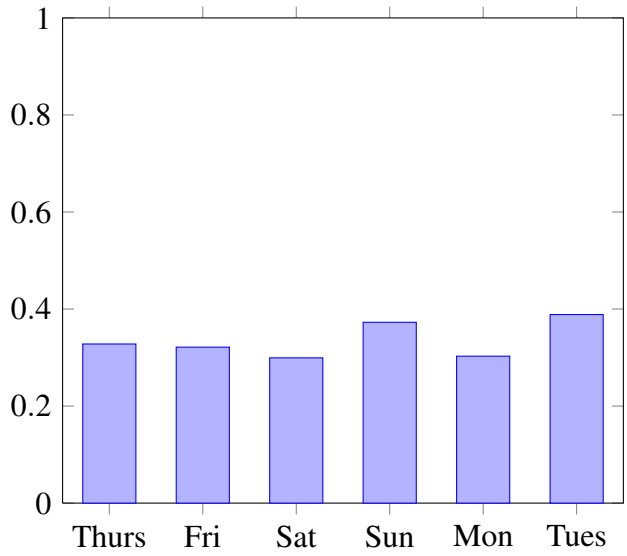

Figure 4.5: Day-to-day accuracy using tf-idf and cosine similarity on src/dst IPs, with test sets filtered to omit source and destination IPs not present in the training sets (May 8-14).

on Friday contains more source IPs than the test set on Saturday, increasing the chance of mis-identified sources. Conversely, the training set on Sunday contains far fewer source IPs than the test set on Monday, so many of the IPs in Monday's set cannot be identified, as our data-mining program never had the chance to learn them.

4.4 Orange: Daily Instances

The rest of the accuracy results were obtained using the Orange [20] data-mining software package. Similar to the unassisted accuracy in §4.3, the training and test sets looked at one day each, aggregating the the number of times each source visited a destination during that day. As discussed in §3.9, accuracies were computed with Orange using Naïve Bayes (NB), Classification Tree, and K-Nearest Neigbors (KNN). We conducted analyses on complete or *unfiltered* sets; on sets that were filtered to remove short-lived flows and DNS queries; and on sets that were filtered to retain only long-lived flows and DNS queries. We applied the filters to the sets in various ways, as described in §4.2. Unlike in §4.3, when building the test sets, we did not consider whether source or destination IPs were in the training sets. That is, we did not omit any IPs from the test set (for any of the following analyses), based on whether those IPs were present in the training set.

mean accuracy: unfiltered sets						
Analysis	*Data*	*Avg # of srcs*	*Avg # of dsts*	*NB*	*Tree*	*KNN*
unfiltered	flow	443	9,046	27.1	8.5	13.5
	dns	480	27,941	46.5	20.3	30.9
	dns (mod)	480	12,389	43.4	20.7	40.0

Table 4.2: Mean accuracy (percent) and mean number of sources and destinations for unfiltered training and testing sets during the week of May 8-14.

4.4.1 Unfiltered Daily Training and Test Sets

For the unfiltered analysis, we used a full day of packet captures for the training set, and tested on the following 24 hours. The training and tests set were not filtered for short-lived destination IP or hostnames, nor was the test set filtered for sources or destinations that were absent from the training set. Both training and test sets contain arrays for every source and destination of every TCP SYN packet captured during their respective 24 hours. We performed the procedure of "train on one day, test on the next day" for the week of May 8-14, 2013.

Destination IPs. For profiles built using destination IPs, Orange's Naïve Bayes module usually returned a higher identification accuracy than the Tree or KNN classifiers (Figure 4.6). The increase in accuracy on Sunday (32.4% for NB) may be due to the fact that most students do not utilize the academic buildings over the weekend. Thus, the activity on Saturday and Sunday would likely be due to a smaller group of the individuals present on both days. Conversely, the considerable decrease in NB accuracy on Monday (13.3%) is likely due to training on fewer source IPs on Sunday, followed testing on a much larger number of sources on Monday, when students return to school. Mean accuracies and numbers of sources and destinations for the week are shown in Table 4.2.

Hostnames. Profiles composed of hostnames from DNS queries yielded higher identification accuracies than profiles using destination IPs, with a maximum NB accuracy of 56.6% and an average accuracy of 46.5% (see Figure 4.7 and Table 4.2). As when using destination IPs, NB generally yielded higher results than Tree or KNN. The NB accuracy graphs in Figure 4.7 display a similar trend as when dst IPs were used, increasing on Sunday with a significant drop on Monday, and then recovering on Tuesday.

When the DNS hostnames were modified (§3.4) prior to analysis, the daily NB accuracies were slightly lower, with an average for the week of 43.4%. However, the KNN averages were

significantly higher when using modified hostnames, increasing by 10% to a weekly average of 40.0%, and registering the highest single-day accuracy of any classifier for the unfiltered tab files, at 63.2% when testing on Sunday.

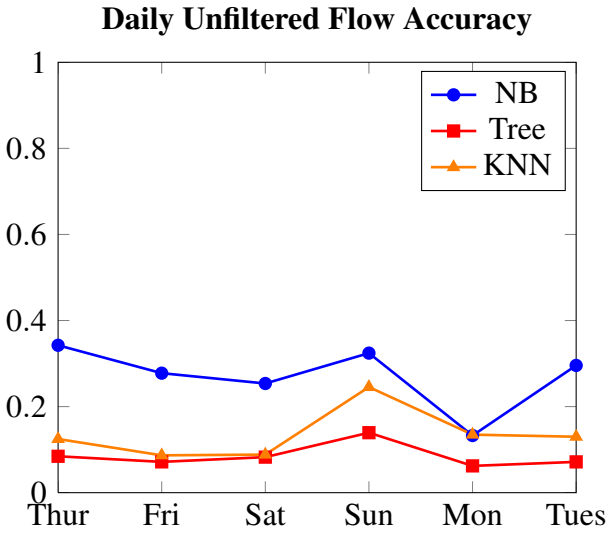

Figure 4.6: Day-to-day Accuracy using unfiltered dst IPs (May 8-14).

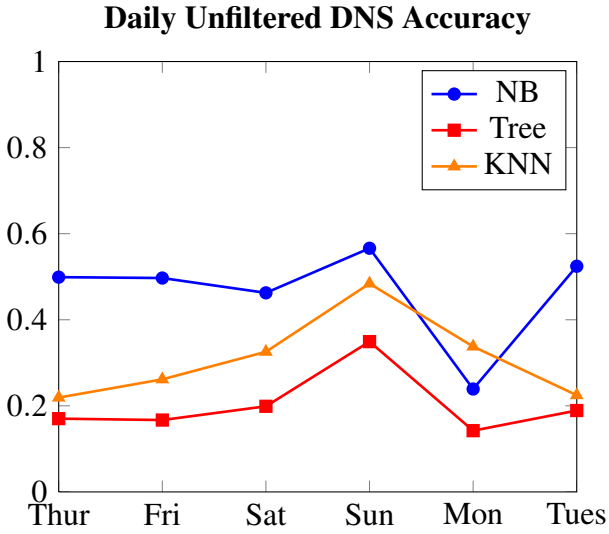

Figure 4.7: Day-to-day Accuracy using unfiltered DNS queries (May 8-14).

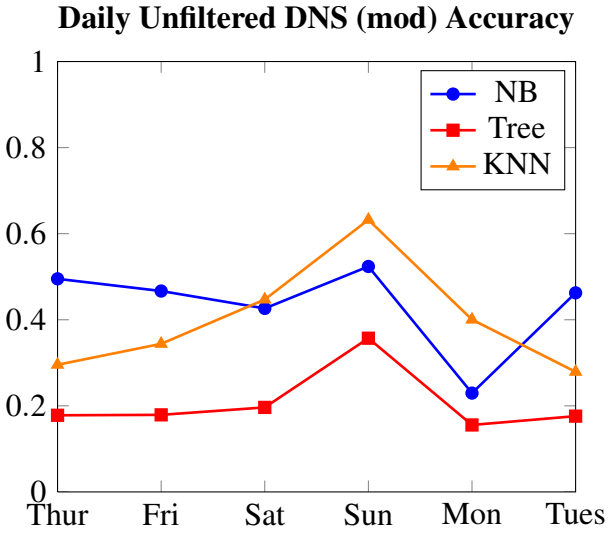

Figure 4.8: Day-to-day Accuracy using unfiltered modified DNS queries (May 8-14).

4.4.2 Filtering Out Short-lived Activity

For this analysis, we used the filters for short-lived accesses (discussed in §4.2) to omit all short-lived network flows and DNS queries during the observed period. As discussed in §4.1.1,

mean accuracy: short-lived filter								
Analysis	*Data*	*Type of Filter*	**Avg # of srcs*	**Avg # of dsts*	*Set(s) Filtered*	*NB*	*Tree*	*KNN*
SL	flow	src/dst	433	5,584	train	29.6	10.3	17.8
					train+test	33.8	11.0	18.6
		dst-only	443	7,511	train	27.0	8.7	13.8
					train+test	26.9	8.9	13.8
SL	dns	src/host	475	13,363	train	47.0	22.7	39.2
					train+test	52.0	23.7	39.2
		host-only	480	27941	train	46.2	22.4	32.8
					train+test	46.6	22.3	32.9
SL	dns (mods)	src/host	476	5,483	train	45.4	21.1	44.8
					train+test	47.6	21.5	44.8
		host-only	479	8,059	train	43.4	19.2	41.1
					train+test	43.6	19.1	41.3

Table 4.3: Mean daily accuracy after filtering out SL accesses (May 8-14). *Average numbers of source and destination IPs are for filtered sets.

the threshold to be considered short-lived was based upon visual inspection of the CDFs, and included the src/dst pairs that contributed to the part of the CDF that appeared to cling to the y-axis. The values for each type of analysis (src/dst, dst-only, src/host, host-only) range from one to six minutes (see Table 4.1).

There were separate filters for SL src/dst pairs, src/hostname pairs, dst IPs only, and hostnames only. We first computed the identification accuracy when just the training sets were filtered, but the test sets remained unfiltered. We then repeated the analysis when both the training and testing sets had short-lived activity filtered out.

Destination IPs. When using the short-lived filters for flow data, we saw the largest increase in accuracy when the filter for SL src/dst pairs was applied to both the testing and training sets. When doing so, the NB accuracies increased to 33.8%, an increase of 6.7% (Figure 4.10). Tree accuracy improved by 2.5%, and KNN improved by 5.1%. When the filter was applied to just the training set (Figure 4.9), the accuracies improved a bit over the unfiltered accuracies, but the increase was not as large. Mean results for the week are shown in Table 4.3. When filtering based on SL src/dst pairs, the mean number of sources and destinations decreased from 443 to 433. This implies that 10 sources never accessed the same website past one short user session. Surprisingly, when the filter was set to only filter out destination IPs irrespective of the source IP

address, it did not seem to matter much whether the test set was filtered, or just the training set was filtered. In some cases, the mean accuracy actually decreased, albeit marginally (Figures 4.11 and 4.12).

Hostnames. When filtering for short-lived src/hostname pairs, accuracy improved the most when filtering both train/test for src/hostnames (as with the flow data), with NB increasing 5.5% to 50.0% and KNN improving by 8.3% (Figure 4.14). As when using flow data, the other applications of the SL filter to DNS data yielded only minor (and sometimes worse) accuracies. When hostnames irrespective of source were filtered, it again did not matter much whether the test set was filtered (Figures 4.13, 4.15, 4.13).

The effect for the modified DNS hostnames was similar to unmodified ones, except that the increases in accuracy were more modest, with a 4.2% increase in NB accuracy and 4.8% increase for KNN. NB was again the most accurate classifier, though only a 2.8% higher than KNN (See Tables 4.17, 4.18, 4.19, 4.20).

Implications. For both DNS queries and flow accesses, filtering out the extremely short-lived accesses and DNS queries that occurred only during one session, but which account for 50-60% of SYNs and DNS queries, led to improved identification accuracy of roughly 5% for NB and KNN. Thus, it would seem that over half of all user network activity (in our dataset) was, in fact, noise that degraded identification accuracy.

It is practical to filter the training set, as we could obtain it prior to testing on the test set. Filtering the test set, however, may not be practical, as we may not know whether it can be filtered. For our tests, we were curios to see the effect of filtering both the training and test sets, and we supposed that we had access to the test set in advance, i.e., we were not conducting our analysis in real time.

Additionally, it makes sense that the accuracies would be higher with src/dst filter on both the training and test sets, vice just filtering the training sets. When just the training set is filtered, there are potentially many IPs or hostnames that present in the test set, but do not contribute meaningful data. Due to the short-lived nature of these accesses, we don't expect them to appear in both the training and test sets.

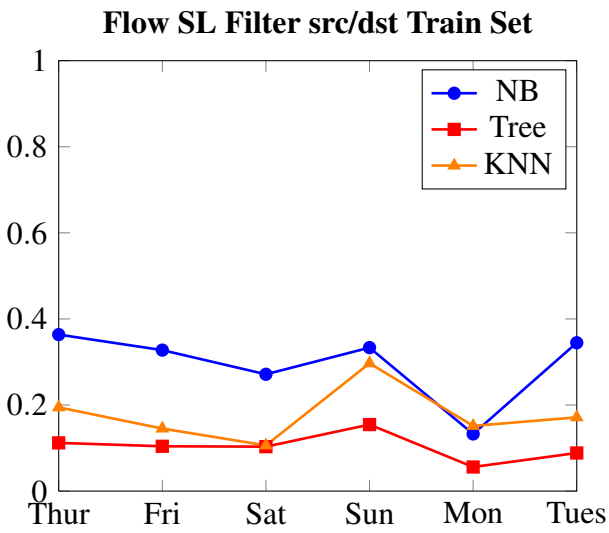

Figure 4.9: Day-to-day Accuracy (May 8-14) filtering training set for src/dst pair sliding-window lifetimes < 371 sec, the bottom 50% of accesses.

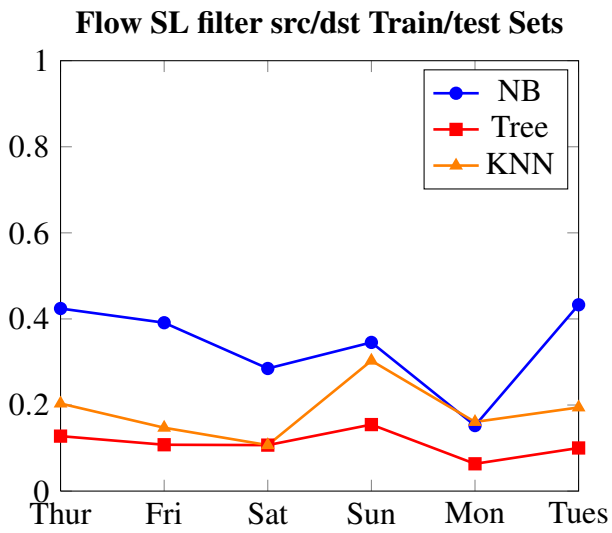

Figure 4.10: Day-to-day Accuracy (May 8-14) filtering train/test sets for src/dst pair sliding-window lifetimes < 371 sec, the bottom 50% of accesses.

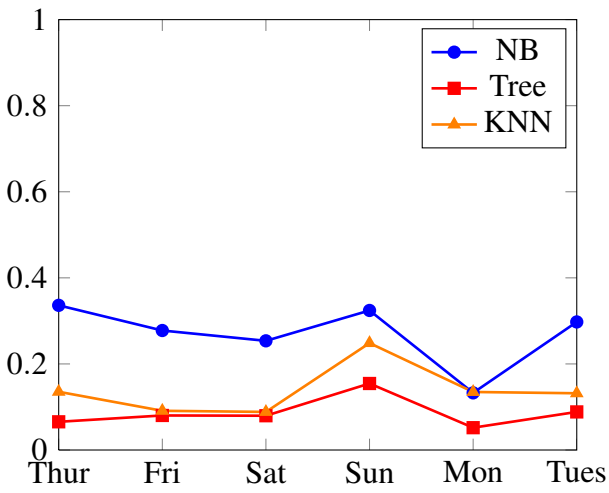

Figure 4.11: Day-to-day Accuracy (May 8-14) filtering training set for dst-only sliding-window lifetimes < 653 sec, the bottom 40% of accesses.

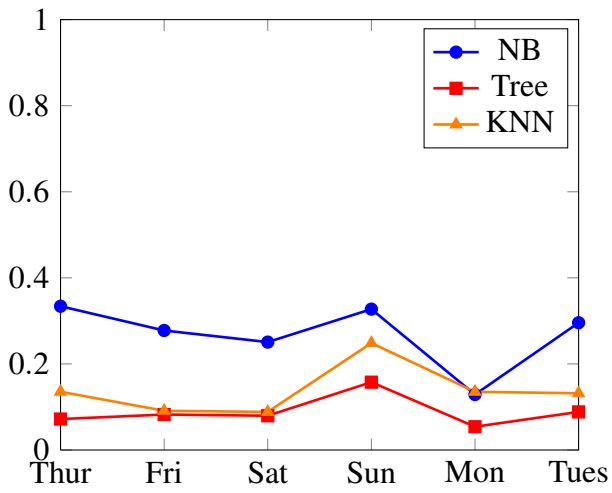

Figure 4.12: Day-to-day Accuracy (May 8-14) filtering train/test sets for dst-only sliding-window lifetimes < 653 sec, the bottom 40% of accesses.

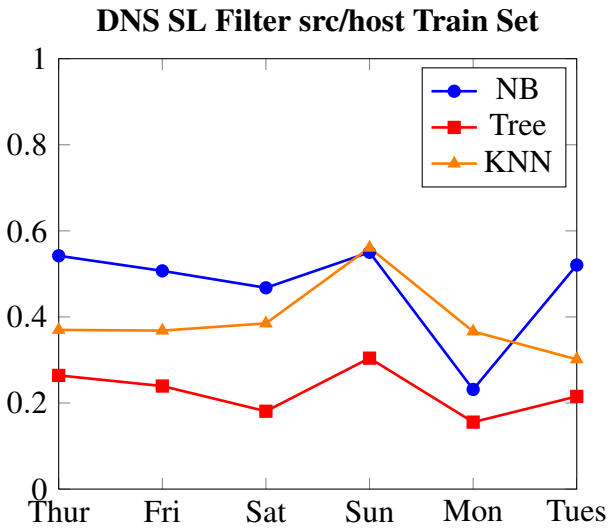

Figure 4.13: Day-to-day Accuracy (May 8-14) filtering training set for src/hostname pair sliding-window lifetimes < 1446 sec, the bottom 60% of accesses.

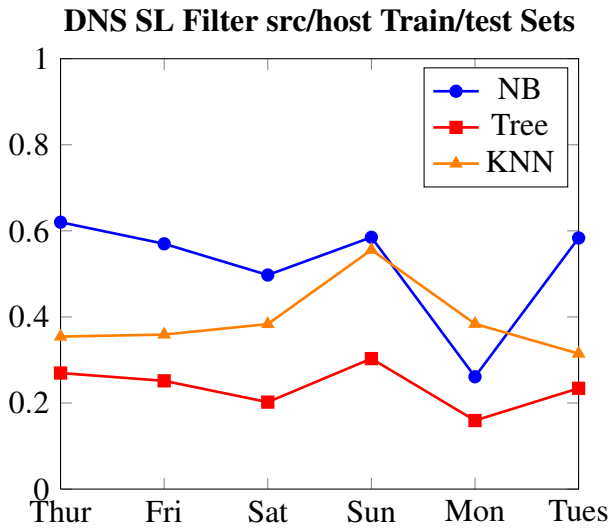

Figure 4.14: Day-to-day Accuracy (May 8-14) filtering train/test sets for src/host pair sliding-window lifetimes < 1446 sec, the bottom 60% of accesses.

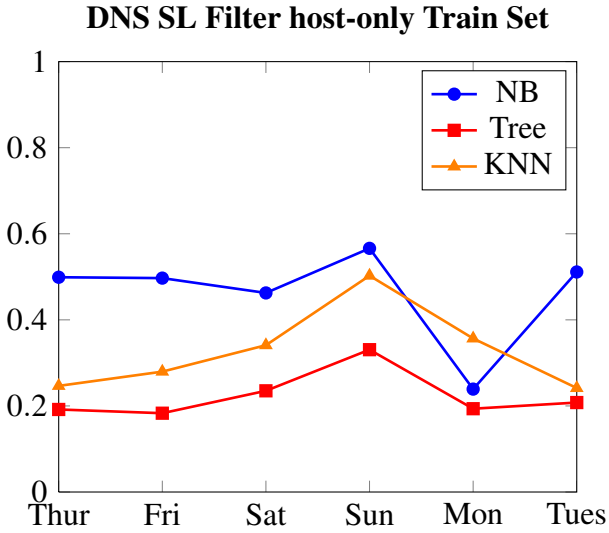

Figure 4.15: Day-to-day Accuracy (May 8-14) filtering training set for host-only sliding-window lifetimes < 97 sec, the bottom 57% of accesses.

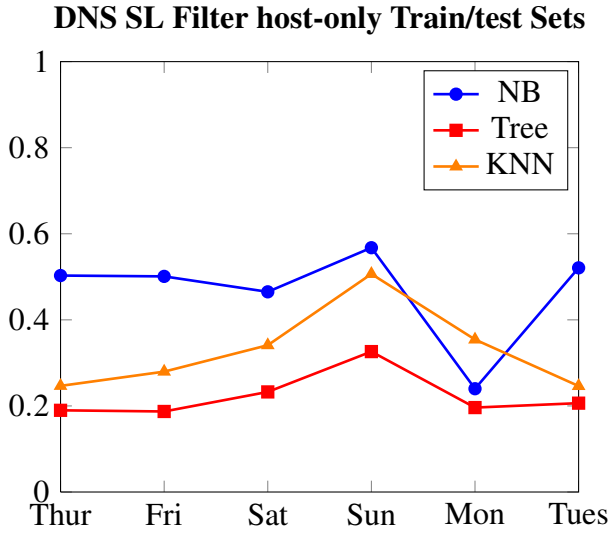

Figure 4.16: Day-to-day Accuracy (May 8-14) filtering train/test sets for host-only sliding-window lifetimes < 97 sec, the bottom 57% of accesses.

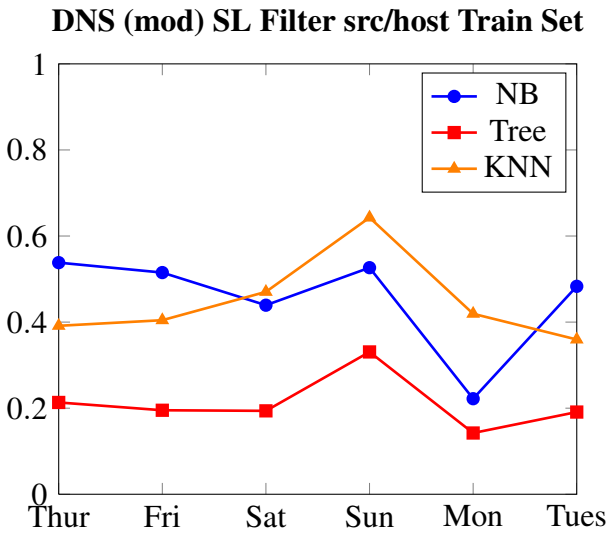

Figure 4.17: Day-to-day Accuracy (May 8-14) filtering modified training set for src/hostname pair sliding-window lifetimes < 88 sec, the bottom 50% of accesses.

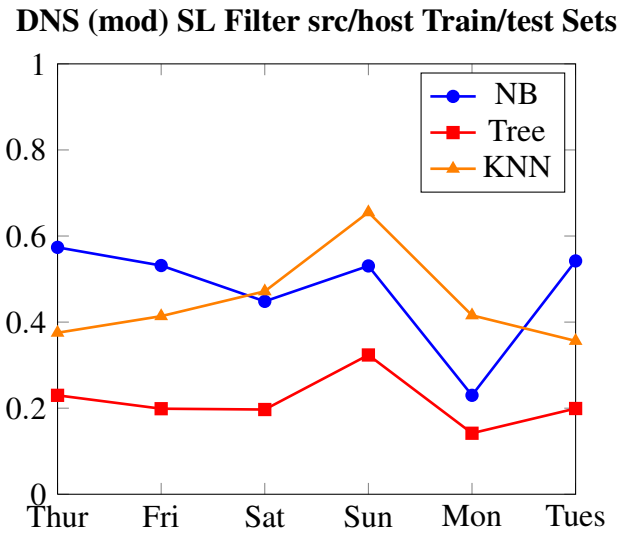

Figure 4.18: Day-to-day Accuracy (May 8-14) filtering modified train/test sets for src/host pair sliding-window lifetimes < 88 sec, the bottom 50% of accesses.

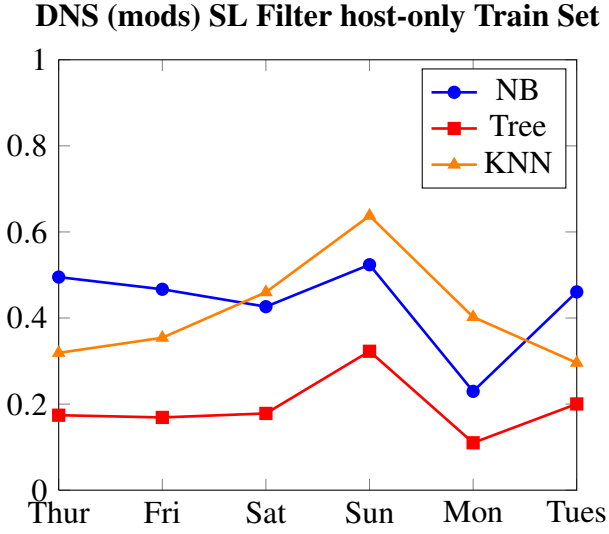

Figure 4.19: Day-to-day Accuracy (May 8-14) filtering modified training set for host-only sliding-window lifetimes < 62 sec, the bottom 60% of accesses.

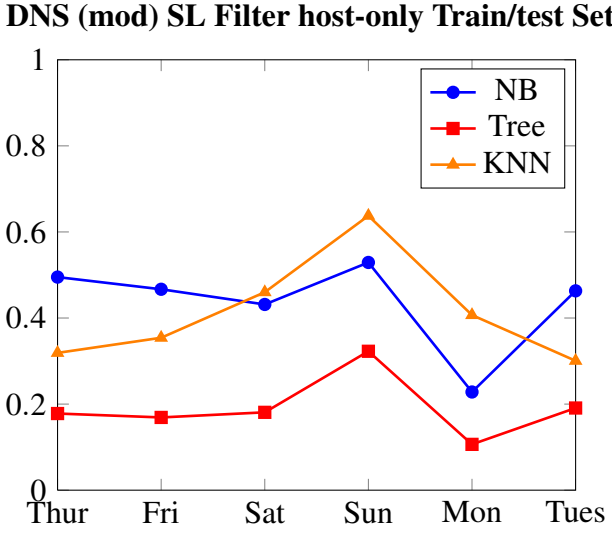

Figure 4.20: Day-to-day Accuracy (May 8-14) filtering modified train/test sets for host-only sliding-window lifetimes < 62 sec, the bottom 60% of accesses.

4.4.3 Filter to Retain only Long-lived Activity

The long-lived analysis was performed in a similar manner as the short-lived analysis in §4.4.2, except that the filters were constructed to retain (vice omit) src/dst or src/hostname pairs, or

mean daily accuracy with 6-day filter								
Analysis	*Data*	*Type of Filter*	**Avg # of srcs*	*dsts*	*Set(s) Filtered*	*NB*	*Tree*	*KNN*
LL-6d	flow	src/dst	366	2,213	train	15.2	7.0	18.2
					train+test	34.8	17.5	34.5
		dst-only	441	4,687	train	26.8	9.5	15.3
					train+test	26.7	9.6	15.4
LL-6d	dns	src/host	434	4,766	train	31.0	17.2	42.6
					train+test	59.4	33.6	59.5
		host-only	478	11,369	train	45.8	20.3	34.8
					train+test	46.4	20.9	35.2
LL-6d	dns (mod)	src/host	437	1,797	train	30.5	14.4	43.4
					train+test	54.6	28.8	63.1
		host-only	479	4,308	train	43.5	19.3	42.4
					train+test	44.0	19.3	42.6

Table 4.4: Mean daily accuracy (percent) after filtering to keep LL accesses with sliding-window lifetimes longer than 6 days (May 8-14). *Average numbers of sources and destinations are for the filtered sets.

individual dsts or hostnames, that are in the filter list. We calculated the accuracies when first using six days as the minimum lifetime to consider accesses as long-lived. We then conducted tests using a five-day threshold to consider accesses to be long-lived, and compared the results. All mean accuracy results are shown in Tables 4.4 and 4.5.

Flow Data (Lifetime > 6 Days). Interestingly, the identification accuracy for long-lived flows with a lifetime greater than six days had a mixed record across NB, Tree, and KNN (see Table 4.4. The NB accuracies increased only when filtering both training and test sets to retain LL src/dst pairs, and that increase was only slightly better than when filtering out SL accesses (34.8% for LL, and 33.8% for SL). When filtering just the training set to retain LL src/dst pairs, the accuracy decreased from the SL analysis, from 29.6% to 15.2%. It is possible that our LL filter is removing accesses that are relevant on at least a day-to-day scale. Additionally, it could be that our lifetime calculation of *last seen time - first seen time* may be "rewarding" infrequent accesses that happened to be accessed six days apart, but were not accessed on adjacent days. As with the SL analysis, the LL results for Tree and KNN showed significant improvement when filtering both train/test data sets using src/dst pairs (Figure 4.22). The KNN accuracy was almost the same score as NB (34.5% versus 34.8%). This was a 21% increase for KNN over the unfiltered accuracies. Additionally, the KNN accuracies when filtering train/test sets to retain

only LL src/dst pairs were 15.9% higher than when filtering to omit just SL src/dst pairs (34.5% compared to 18.6%). When filtering just the training set, KNN increased by less than half a percent over the SL analysis.

The price of increased accuracy was 77 fewer sources when the six-day LL filter was used. Again there was no significant effect when filtering to keep just destination IPs, regardless of whether the test set was filtered.

DNS Data (Lifetime > 6 Days). Similar to results using flow data, a six-day filter on the DNS data sets saw the highest identification accuracy with NB and KNN virtually tied, at 59.4% and 59.5%, respectively, when filtering both train/test sets (Figure 4.30). The increase for NB over the SL analysis was 7.4%, and the increase for KNN was 20.2%. When just the training set was filtered for src/dst pairs, the NB accuracy was 31.0%, a 16% decrease from the SL analysis, and KNN registered a 3.4% increase. For hostname-only filtering, it again did not seem to matter much whether the the test set was filtered along with the training set.

With the modified DNS data, when filtering both train/test sets, the NB score was about 5% lower than the unmodified DNS hostnames. But the KNN graph was noticeably higher than the unmodified hostnames, yielding the highest mean accuracy of all the classifiers at 63.1%, and also had the highest one-day accuracy of 76.3% for the test on Sunday. The accuracies for NB and Tree were each about 7% higher than the SL analysis, at 54.6% for NB and 28.8% for Tree.

When just the training set was filtered for src/dst pairs, the modified DNS data plunged by 14.9% for NB (to 30.5%) as compared the SL analysis, and Tree decreased by 6.7% to 14.4.%. KNN was roughly the same as when filtering SL src/host pairs.

Flow Data (Lifetime > 5 Days). We next calculated the identification accuracies using a threshold of five days, rather than six days, for accesses to be considered long-lived. The thinking behind this was that perhaps six days was a bit stringent, as there may be some accesses that a user routinely makes, but were not observed six days after the first access.

The results when using a five-day lifetime were slightly lower than when using lifetimes of six days. For filtering both training and testing sets on flow data, NB accuracy decreased by 2.3% compared to the six-day filter, and KNN decreased by 6.4%. NB was about the same as the SL analysis, but KNN increased by 9.5%, reaching 28.1%. When filtering just the training set for src/dst pairs, the NB accuracy was 18.8%, which was 10.8% lower the when filtering out SL accesses, and 3.6% higher than for a six-day lifetime. For KNN accuracies, when filtering just

mean daily accuracy with 5-day filter								
Analysis	*Data*	*Type of Filter*	**Avg # of srcs*	**Avg # of dsts*	*Set(s) Filtered*	*NB*	*Tree*	*KNN*
LL-5d	flow	src/dst	398	3,038	train	18.8	9.4	20.5
					train+test	32.5	14.2	28.1
		dst-only	442	5,935	train	27.0	8.7	14.1
					train+test	26.9	8.7	14.0
LL-5d	dns	src/host	456	6,808	train	45.9	20.0	34.5
					train+test	57.2	29.9	52.4
		host-only	478	15,133	train	45.9	20.0	34.5
					train+test	46.4	19.9	34.3
LL-5d	dns (mod)	src/host	459	2,665	train	37.6	18.3	46.5
					train+test	50.6	26.3	55.7
		host-only	479	5,939	train	43.2	20.0	41.7
					train+test	43.7	20.1	41.8

Table 4.5: Mean daily accuracy (percent) after filtering to keep LL accesses with sliding-window lifetimes longer than 5 days (May 8-14). *Average numbers of sources and destinations are for the filtered sets.

the training sets, the accuracy increased by 2.3% over the six-day LL filter, and and was 2.7% higher than the SL analysis. See Figures 4.25, 4.26, 4.27, and 4.28.

DNS Data (Lifetime > 5 Days). For DNS data, the NB accuracy when filtering train/test sets for src/dst pairs was 57.2%, which was slightly lower than for the six-day filter, and 5.2% higher than the SL analysis. For KNN, the accuracy was 52.4%, which was 7.1% less than the six-day filter, and 13.2% higher than the SL analysis. When filtering just the training set, NB 1.1% lower than the SL analysis, though it was improved by 14.9% over the six-day filter. For the modified DNS data, the trends were similar, though the values were different. See Figures 4.33 through 4.42.

Implications. The long-lived filter yielded larger increases in accuracy than the SL filter when both the training and test sets were filtered, which make sense. For destination IPs or hostnames to be retained by the LL filter, the access or query had to occur at least twice during the observed period. In contrast, the SL accesses were inherently limited to one session, and so would not be present in both a training set and test set. When testing or training on a day in which one of the SL flows or hostnames is not seen, the effect is to create more "noise" that hinders accuracy. Of note, when filtering just the training sets, the LL filters often fared worse than the SL filters. In

the SL analysis, filtering just the training set would have meant a training set with none of the SL accesses, but all other accesses, including the LL and in-between accesses. With the LL filters, however, the in-between accesses were no longer present in the training set but, of course, were found in the unfiltered test sets. That the SL filters performed better in this scenario, combined with the fact that the five-day filter often yieled higher accuracies than the six-day filter, seems to imply that the access with lifetimes that are in-between SL and LL are rather important, at least on a day-to-day level.

Also, it can be seen that the LL filter led to larger increases in accuracy for DNS queries than for IP flows, although the relative ratio of flow accuracy to DNS hostname accuracy remained about the same, at 0.586 for the six-day LL filter versus 0.582 for unfiltered data.

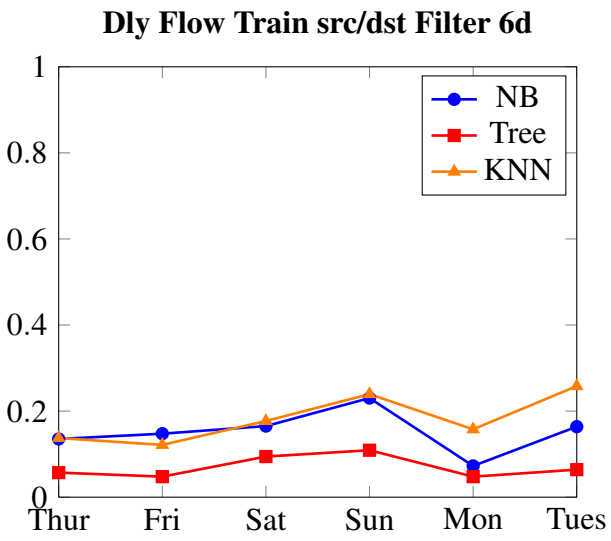

Figure 4.21: Day-to-day Accuracy (May 8-14) filtered training set for src/dst pair sliding-window lifetimes > 6 days.

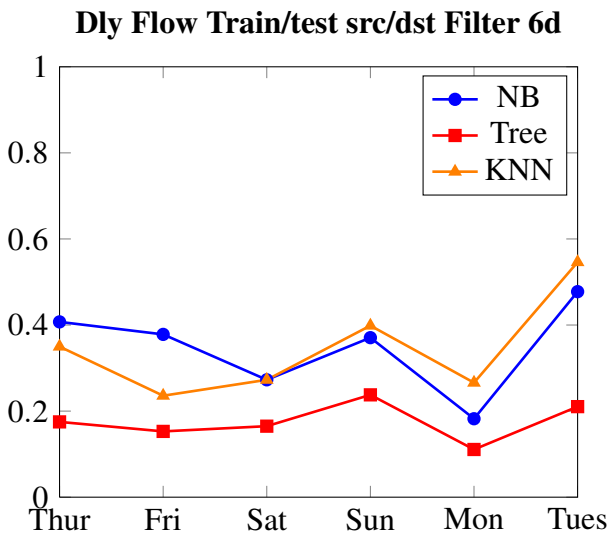

Figure 4.22: Day-to-day Accuracy (May 8-14) filtering train/test sets for src/dst sliding-window lifetimes > 6 days.

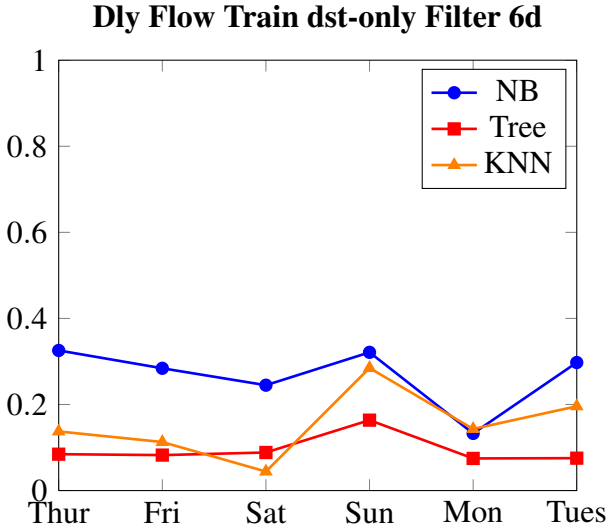

Figure 4.23: Day-to-day Accuracy (May 8-14) filtering training set for dst-only sliding-window lifetimes > 6 days.

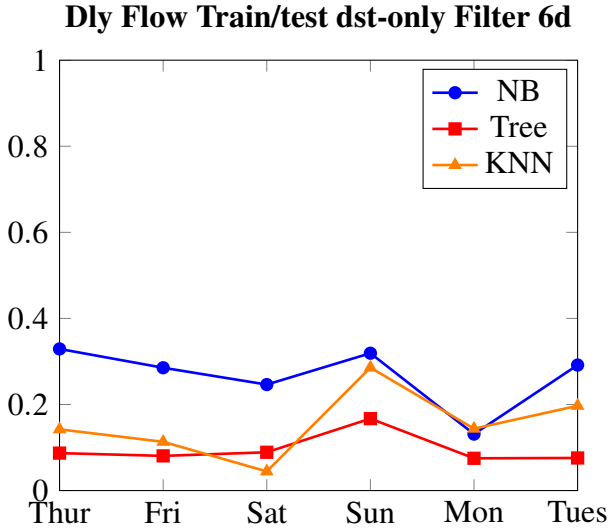

Figure 4.24: Day-to-day Accuracy (May 8-14) filtering train/test sets for dst-only sliding-window lifetimes > 6 days.

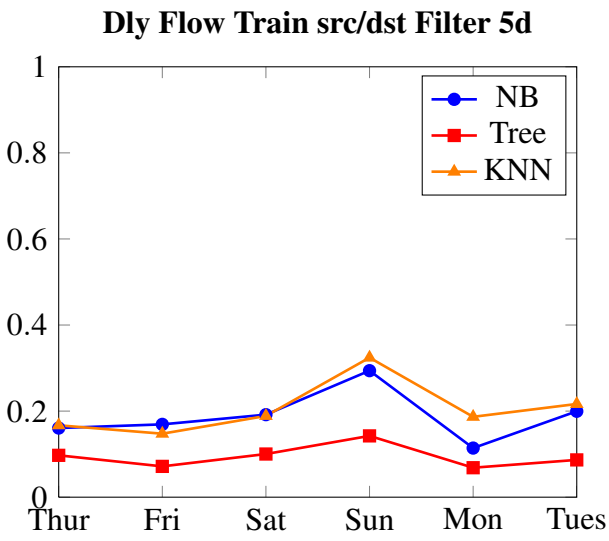

Figure 4.25: Day-to-day Accuracy (May 8-14) filtered training set for src/dst pair sliding-window lifetimes > 5 days.

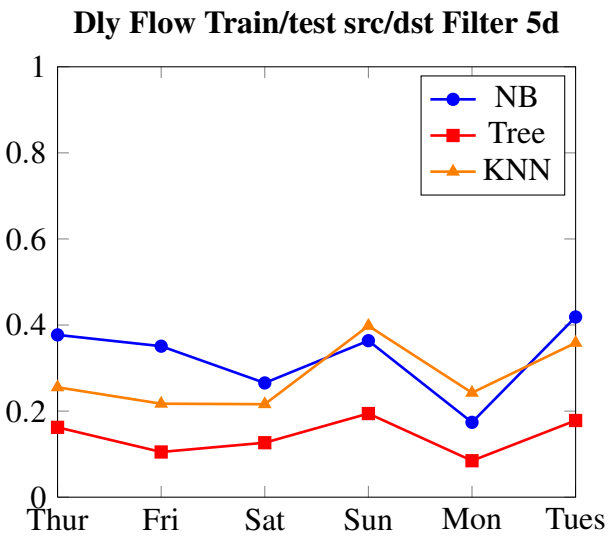

Figure 4.26: Day-to-day Accuracy (May 8-14) filtering train/test sets for src/dst sliding-window lifetimes > 5 days.

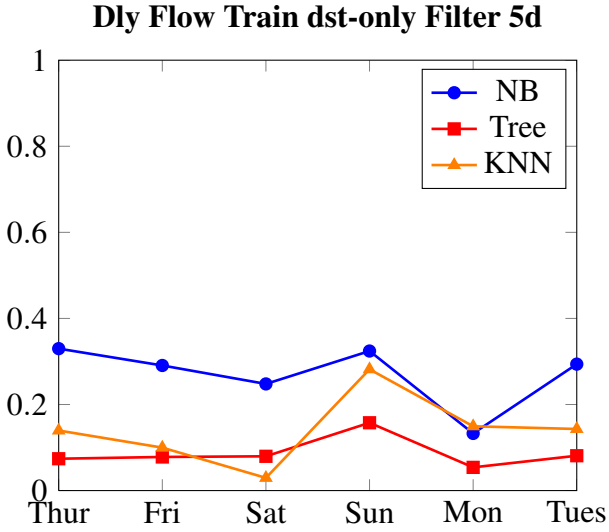

Figure 4.27: Day-to-day Accuracy (May 8-14) filtering training set for dst-only sliding-window lifetimes > 5 days.

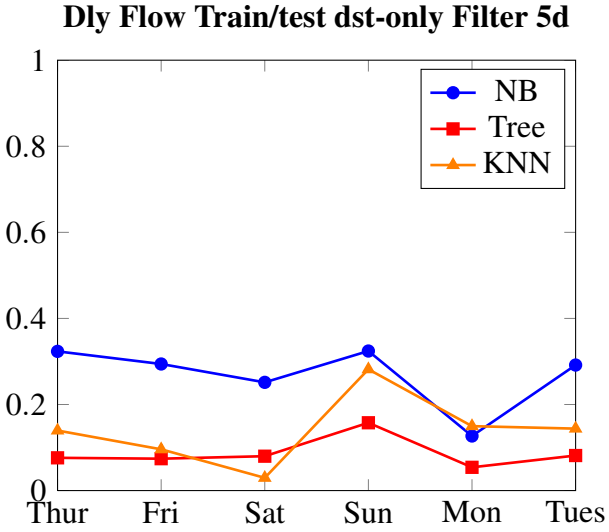

Figure 4.28: Day-to-day Accuracy (May 8-14) filtering train/test sets for dst-only sliding-window lifetimes > 5 days.

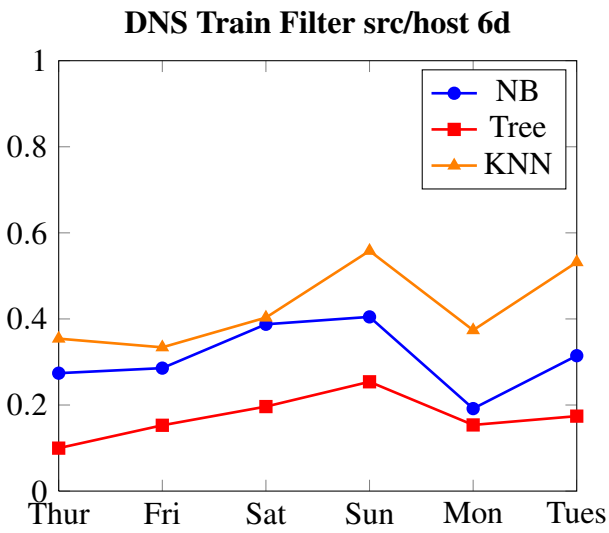

Figure 4.29: Day-to-day Accuracy (May 8-14) filtering training set for src/host pair sliding-window lifetimes > 6 days.

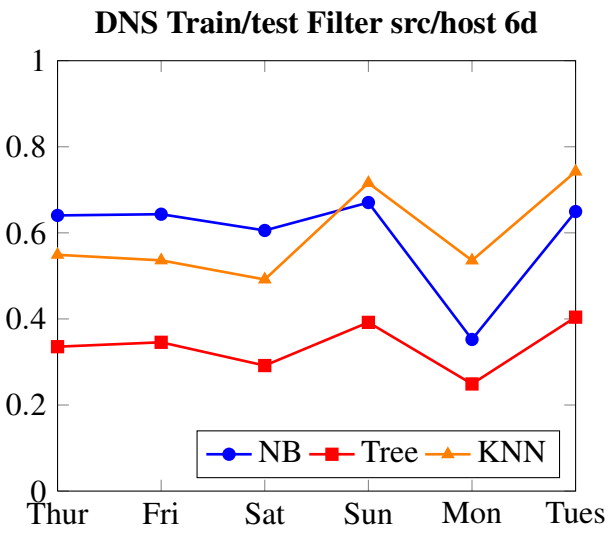

Figure 4.30: Day-to-day Accuracy (May 8-14) filtering train/test sets for src/host sliding-window lifetimes > 6 days.

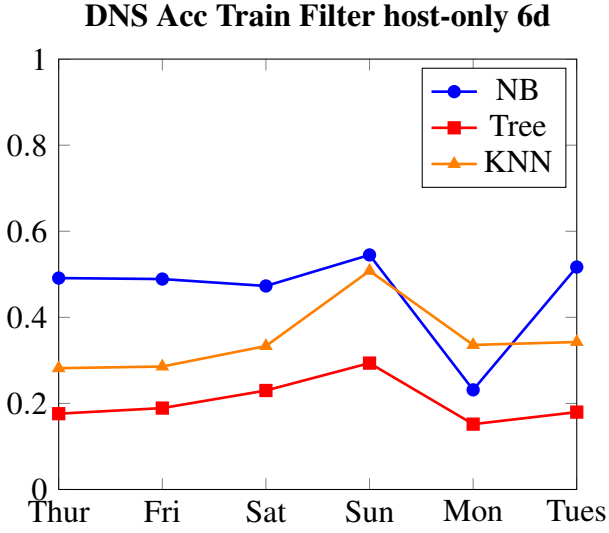

Figure 4.31: Day-to-day Accuracy (May 8-14) filtering training set for host-only sliding-window lifetimes > 6 days.

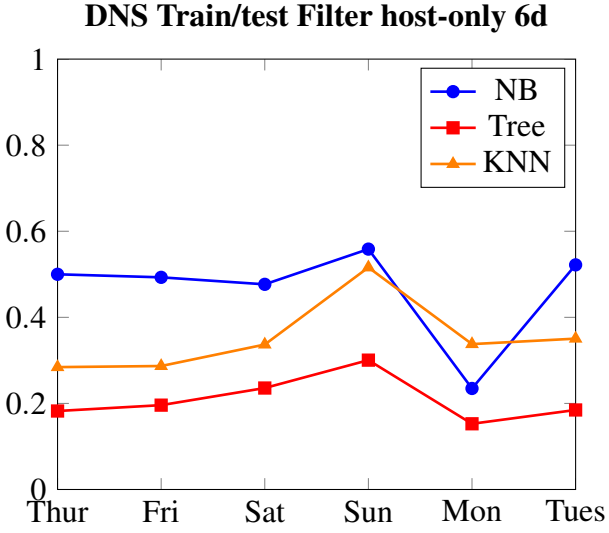

Figure 4.32: Day-to-day Accuracy (May 8-14) filtering train/test sets for host-only sliding-window lifetimes > 6 days.

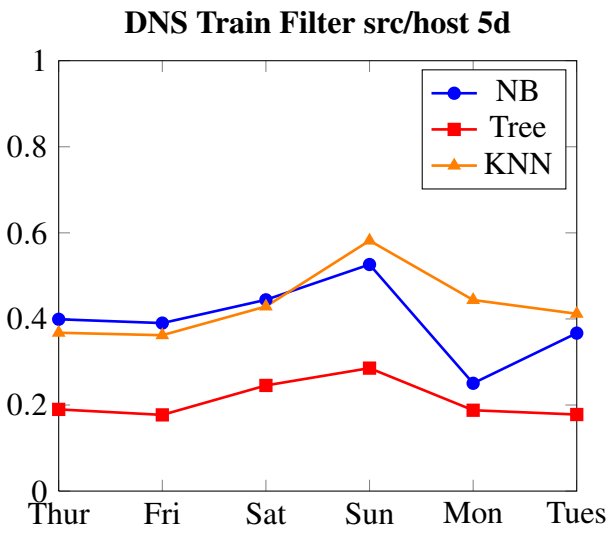

Figure 4.33: Day-to-day Accuracy (May 8-14) filtering training set for src/host pair slidinw-window lifetimes > 5 days.

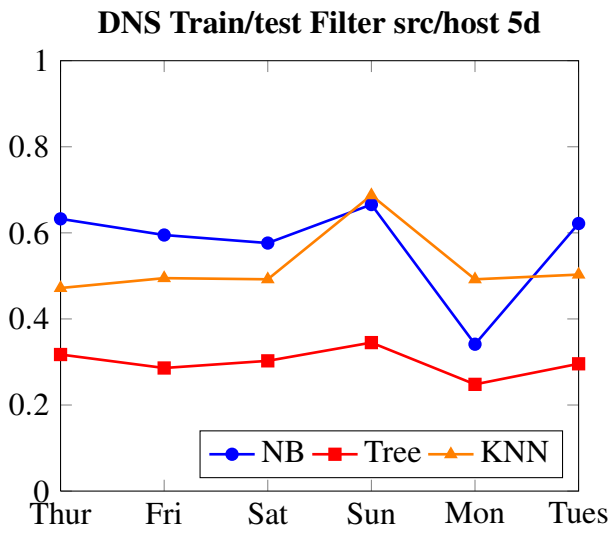

Figure 4.34: Day-to-day Accuracy (May 8-14) filtering train/test sets for src/host sliding-window lifetimes > 5 days.

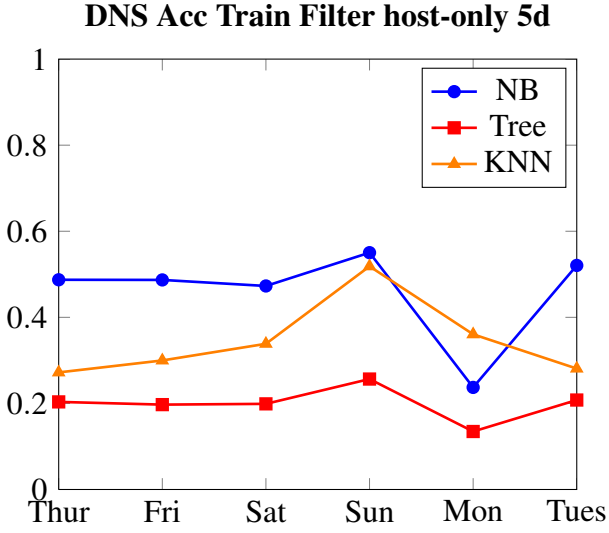

Figure 4.35: Day-to-day Accuracy (May 8-14) filtering training set for host-only sliding-window lifetimes > 5 days.

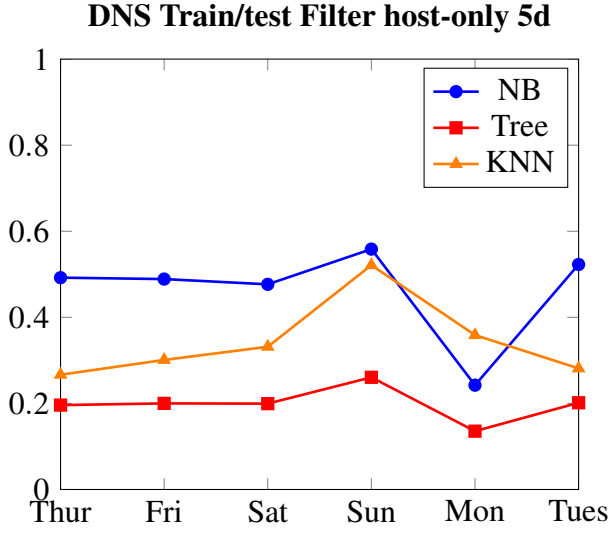

Figure 4.36: Day-to-day Accuracy (May 8-14) filtering train/test sets for host-only sliding-window lifetimes > 5 days.

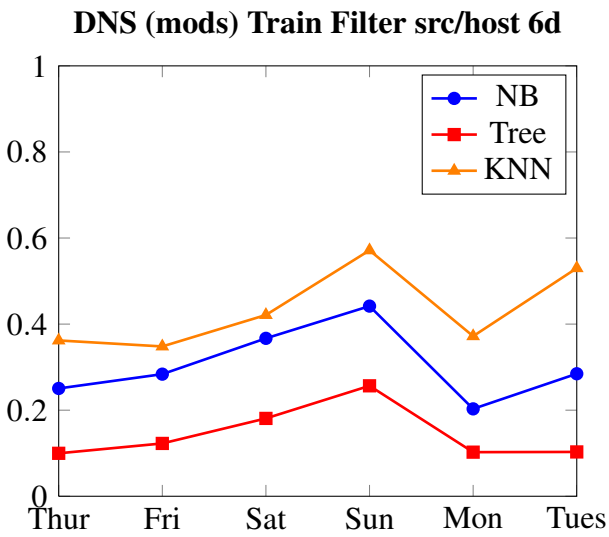

Figure 4.37: Day-to-day Accuracy (May 8-14) filtering modified training set for src/host sliding-window lifetimes > 6 days.

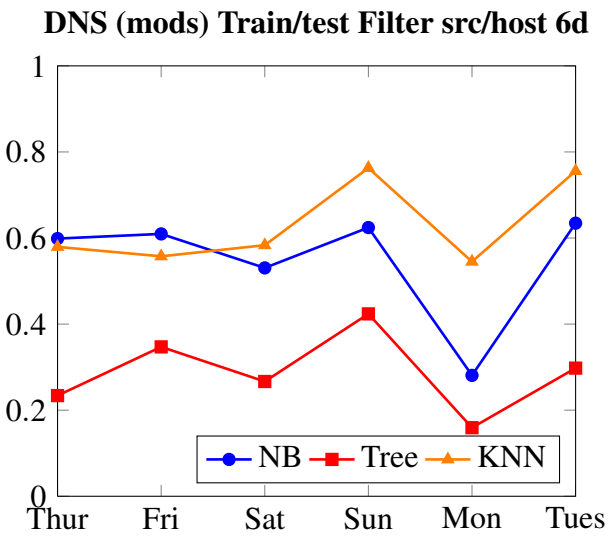

Figure 4.38: Day-to-day Accuracy (May 8-14) filtering modified train/test sets for src/host sliding-window lifetimes > 6 days.

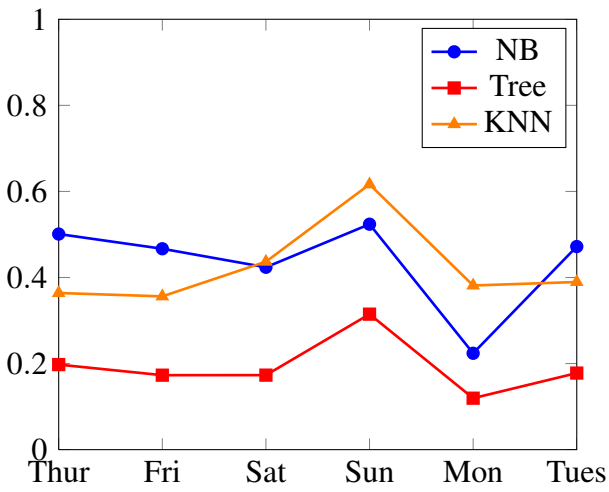

Figure 4.39: Day-to-day Accuracy (May 8-14) filtering modified DNS training set for host-only sliding-window lifetimes > 6 days.

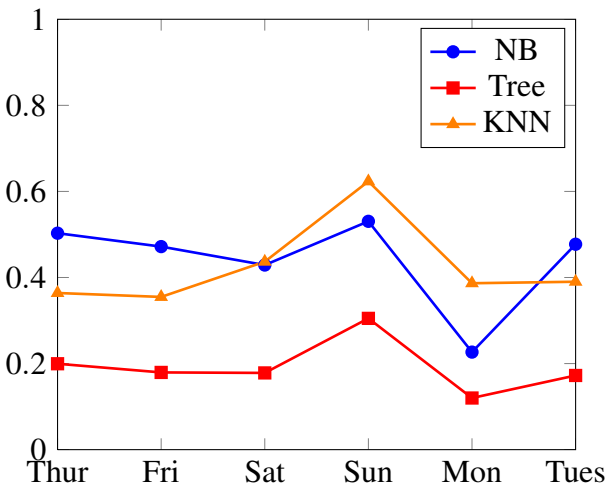

Figure 4.40: Day-to-day Accuracy (May 8-14) filtering modified train/test sets for host-only sliding-window lifetimes > 6 days.

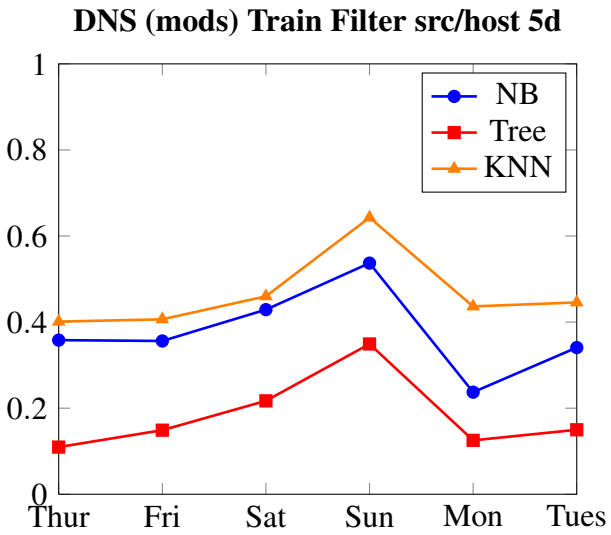

Figure 4.41: Day-to-day Accuracy (May 8-14) filtering modified training set for src/host sliding-window lifetimes > 5 days.

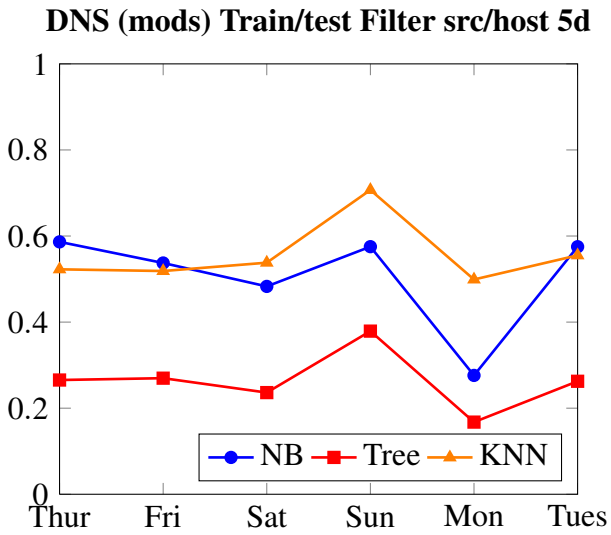

Figure 4.42: Day-to-day Accuracy (May 8-14) filtering modified train/test sets for src/host sliding-window lifetimes > 5 days.

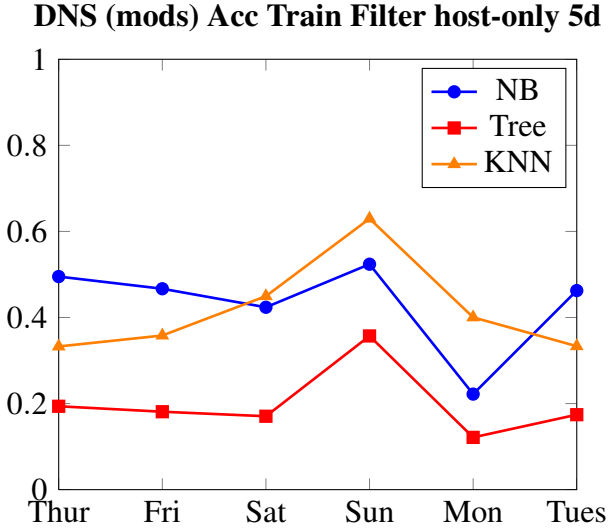

Figure 4.43: Day-to-day Accuracy (May 8-14) filtering training set for modified DNS host-only sliding-window lifetimes > 5 days.

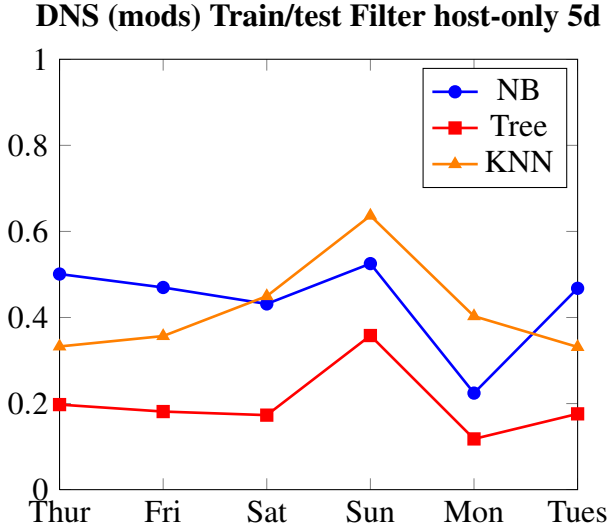

Figure 4.44: Day-to-day Accuracy (May 8-14) filtering modified train/test sets for host-only sliding-window lifetimes >5 days.

4.4.4 Orange: Filter for Accesses Active for at Least X Days

For the *at least X days* tests, we filtered to retain accesses that were active for at least X days during the week of May 8-14, 2013, where X ranged from three to six days. The tab files that are the input for Orange contained a matrix of access frequency, weighted by tf-idf. However,

Flow: mean accuracy for at least X different days							
Analysis	*Type of Filter*	**Avg # of srcs*	*dsts*	*Set(s) Filtered*	*NB*	*Tree*	*KNN*
≥3 diff days	src/dst	404	3,034	train	25.0	10.3	27.7
				train+test	39.0	15.6	36.0
	dst-only	442	5,869	train	27.2	9.3	13.5
				train+test	27.0	9.3	13.6
≥4 diff days	src/dst	381	2,285	train	21.8	9.4	28.1
				train+test	41.1	18.5	43.7
	dst-only	442	4,950	train	27.1	8.3	15.8
				train+test	26.8	8.4	15.9
≥5 diff days	src/dst	343	1,721	train	17.6	11.2	27.3
				train+test	42.7	25.1	52.2
	dst-only	448	4,105	train	26.6	8.5	18.0
				train+test	26.5	8.5	18.2
≥6 diff days	src/dst	289	1,187	train	13.0	10.2	23.2
				train+test	39.4	30.0	54.3
	dst-only	440	2,482	train	25.0	9.7	21.6
				train+test	25.2	9.6	21.8

Table 4.6: Mean daily accuracy (percent) after filtering flow data to keep accesses that were observed on at least 3, 4, 5, or 6 separate days (May 8-14). *Average numbers of sources and destinations are for the filtered sets.

these frequencies are aggregated over the whole testing or training period, and do not reflect whether the accesses were regular visits that might be a marker of usual network behavior, or whether the accesses were part of a short flurry of visits.

For example, suppose during a certain period, a user visited site *A* three times on the first day and three times on the last day (with no visits in between), and also visited site *B* for six days in a row, one time each day. Further suppose that sites *A* and *B* have the same relative frequency across all users. Then, *A* and *B* could register the same tf-idf score, since the two sites have the same total number of visits during this observation period. Additionally, under the previous method using lifetime-based filters, despite being accessed in a rather different manner, both site *A* and *B* would have lifetimes of six days, which could improperly weight their importance. With the *at least X days* method, however, site *A* would register as having been visited on two distinct days, and *B* would have a visit-count of six distinct days, more clearly indicating that *B* is a regularly visited destination. Certainly, for the "train on one day, test on the next day"

DNS: mean accuracy for at least X different days							
Analysis	*Type of Filter*	**Avg # of srcs*	**Avg # of dsts*	*Set(s) Filtered*	*NB*	*Tree*	*KNN*
≥3 different days	src/host	457	7,282	train	44.0	22.0	53.2
				train+test	60.0	29.2	58.2
	host-only	478	14,731	train	45.9	21.2	35.0
				train+test	46.0	20.8	35.1
≥4 different days	src/host	442	5,146	train	41.4	22.9	56.6
				train+test	63.8	33.1	67.7
	host-only	477	11,841	train	46.0	20.4	37.2
				train+test	46.7	20.8	37.5
≥5 different days	src/host	416	3,749	train	37.8	23.5	57.8
				train+test	65.5	39.1	77.8
	host-only	476	9,024	train	45.8	20.9	41.5
				train+test	46.0	21.4	41.9
≥6 different days	src/host	366	1,979	train	32.2	23.2	58.1
				train+test	69.6	44.2	86.5
	host-only	476	4,544	train	45.2	22.1	43.8
				train+test	45.8	22.2	43.8

Table 4.7: Mean daily accuracy (percent) after filtering DNS data to keep accesses that were observed on at least 3, 4, 5, or 6 separate days (May 8-14). *Average numbers of sources and destinations are for the filtered sets.

model, site *A* would not contribute any information to help identify the source, because (unlike site *B*), *A* was never accessed on two consecutive days and, thus, would never be present in adjacent training and testing sets.

Flow Data. When filtering both training and test sets to retain flow accesses that were observed on at least 3 days, the NB accuracy started at 39.0%, which was 4.2% higher than when using the six-day lifetime, and 11.9% larger than with unfiltered data. NB increased by about 2% for a filter of ≥4 days, then increased to a high of 42.7% with a filter of at least five days, before falling to 39.4% with a filter of ≥6 days. The Tree and KNN classifiers steadily increased with each successive filter, with KNN achieving a high of 54.3%, using a filter of ≥6 days. The number of sources in the filtered sets started at a mean of 404 (91% of unfiltered sources) for the "at least three days" test. The mean sources decreased as the filter became more stringent, until reaching 289 mean sources (65% of unfiltered). See Table 4.6 and Figures 4.46, 4.50, 4.54, and 4.58).

DNS (mod): mean accuracy for at least X different days							
Analysis	*Type of Filter*	**Avg # of srcs*	**Avg # of dsts*	*Set(s) Filtered*	*NB*	*Tree*	*KNN*
≥3 diff days	src/host	459	2,768	train	42.1	21.9	53.5
				train+test	53.0	28.3	59.2
	host-only	479	5,690	train	43.4	20.4	43.5
				train+test	43.6	20.5	43.5
≥4 diff days	src/host	444	1,984	train	39.2	19.3	54.9
				train+test	55.5	29.6	66.8
	host-only	479	4,478	train	43.4	20.9	45.7
				train+test	43.7	21.0	45.7
≥5 diff days	src/host	420	1,424	train	36.2	18.7	54.8
				train+test	59.5	33.1	75.8
	host-only	479	3,322	train	43.4	19.9	49.1
				train+test	43.8	19.5	49.1
≥6 diff days	src/host	368	715	train	31.4	18.4	54.9
				train+test	63.8	40.9	85.3
	host-only	478	1,626	train	43.2	20.1	50.7
				train+test	43.4	20.0	50.8

Table 4.8: Mean daily accuracy (percent) after filtering modified flow data to keep accesses that were observed on at least 3, 4, 5, or 6 separate days (May 8-14). *Average numbers of sources and destinations are for the filtered sets.

DNS Data. When using DNS data, the NB accuracy increased with each filter, from 60.0% for ≥3 days, to a high of 69.6% for at least 6 days when applied to src/hostname pairs in both the training and test sets. Tree and KNN also increased with each filter, with KNN reaching a mean accuracy of 86.5% for a filter of ≥6 days. With the modified DNS hostnames, the NB and Tree accuracies were consistently lower than when using the unmodified hostnames, with a best for NB of 63.8%. On the other hand, the identification accuracies using KNN were just slightly under the KNN accuracies for the unmodified DNS queries, with a maximum value of 85.3%. As with the flow data, there was a decrease in the mean number of sources after the filter was applied, although the proportion of sources was higher with DNS data. The largest decrease was with the filter of at least six days, with a mean of 366 sources for DNS (368 for modified DNS), which is roughly 76% of the unfiltered sources (see Figures 4.62, 4.66, 4.70, 4.74, 4.78, 4.82, 4.86, 4.90).

Discussion. The general trend (for both flow data and DNS data) is that, as the filter becomes more strict with the minimum different days on which to observe a flow or hostname, the mean accuracy increases, but the mean number of observed sources decreases. Thus, the method could be described as being more accurate at identifying fewer sources. Another trend is that KNN noticeably replaces NB as the most accurate classifier, achieving not only the highest mean accuracies for the week, but also the highest accuracy for each day of the week. Additionally, as the minimum number of days increases, the accuracy graphs for NB, Tree, and KNN become stratified. We again see that filtering on both the training and test sets using src/dst or src/hostname pairs yields higher accuracies than just filtering the training set. As with the SL and LL filters, when keying off of just the destination IPs or DNS query hostnames, there was only a slight difference between filtering the training set only, or both training and test sets.

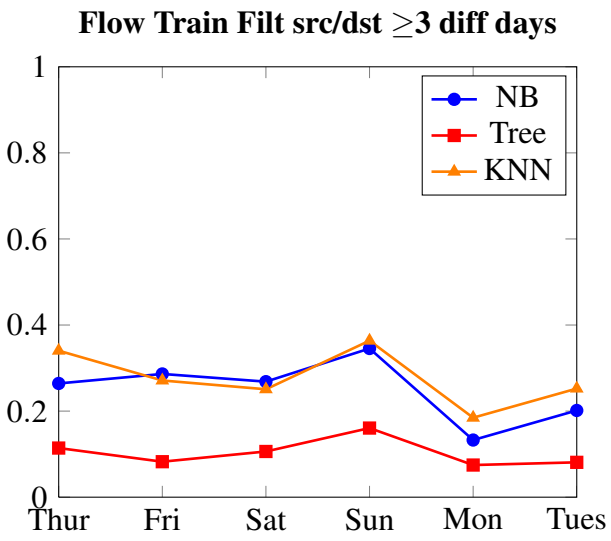

Figure 4.45: Day-to-day Accuracy (May 8-14) filtering training set for src/dst pairs that were seen at least 3 different days during the week.

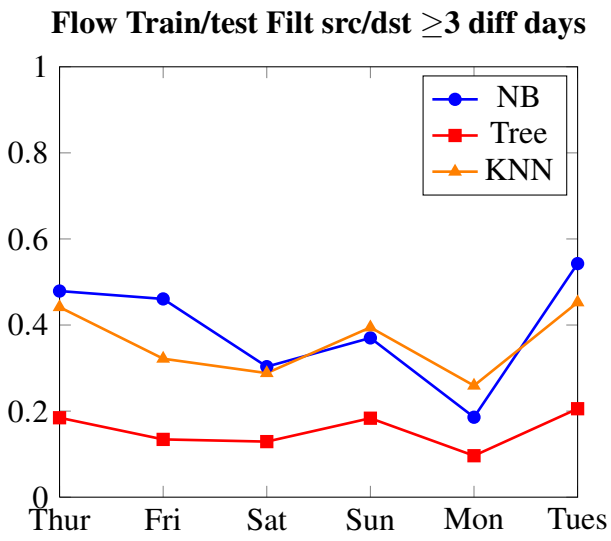

Figure 4.46: Day-to-day Accuracy (May 8-14) filtering training/test sets for src/dst pairs that were seen at least 3 different days during the week.

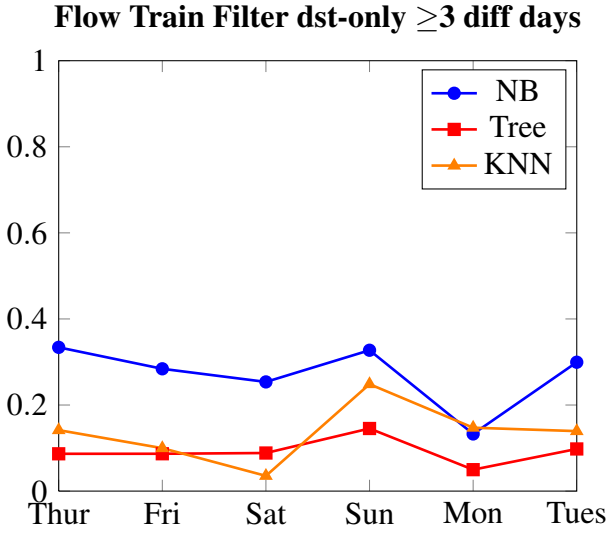

Figure 4.47: Day-to-day Accuracy (May 8-14) filtering training set for dst-only IPs that were seen at least 3 different days during the week.

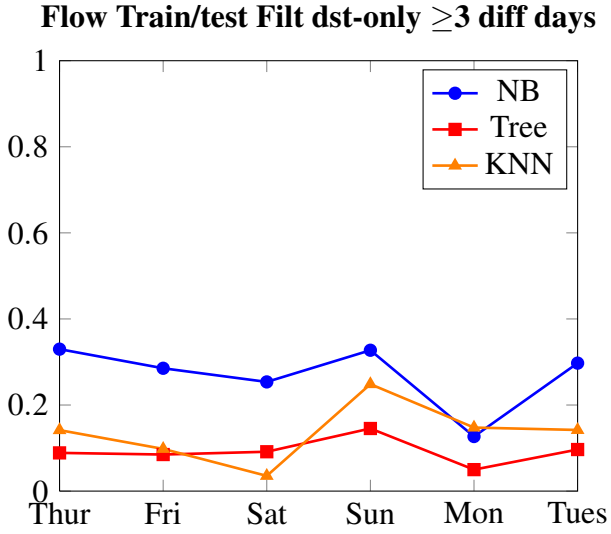

Figure 4.48: Day-to-day Accuracy (May 8-14) filtering training/test sets for dst-only IPs that were seen at least 3 different days during the week.

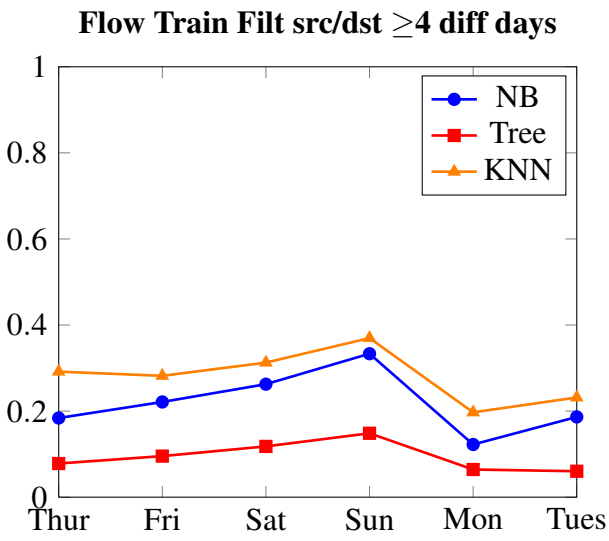

Figure 4.49: Day-to-day Accuracy (May 8-14) filtering training set for src/dst pairs that were seen at least 4 different days during the week.

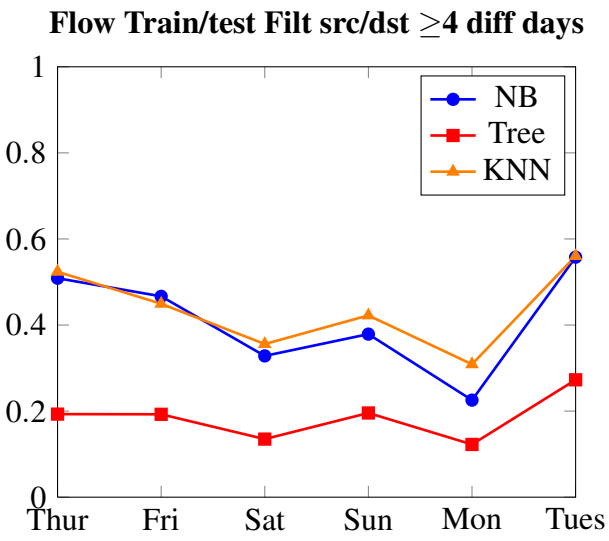

Figure 4.50: Day-to-day Accuracy (May 8-14) filtering training/test sets for src/dst pairs that were seen at least 4 different days during the week.

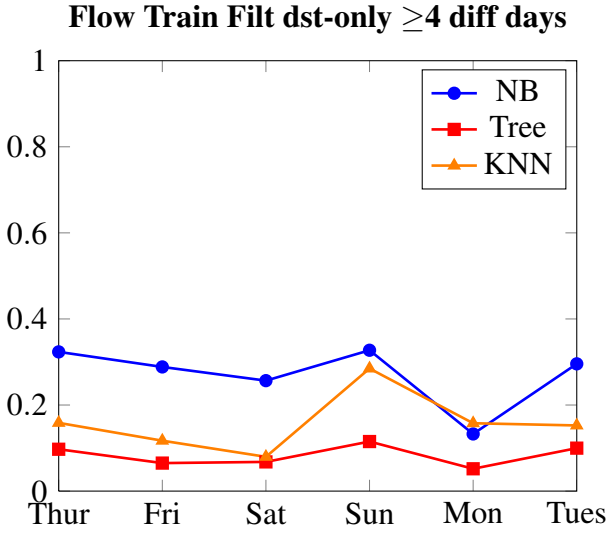

Figure 4.51: Day-to-day Accuracy (May 8-14) filtering training set for dst-only IPs that were seen at least 4 different days during the week.

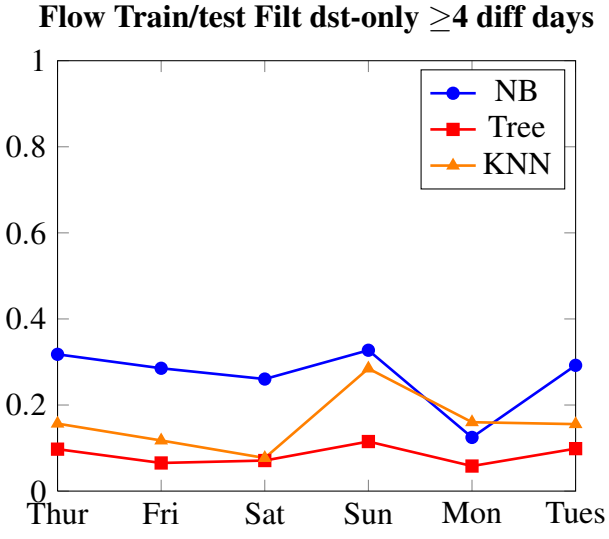

Figure 4.52: Day-to-day Accuracy (May 8-14) filtering training/test sets for dst-only IPs that were seen at least 4 different days during the week.

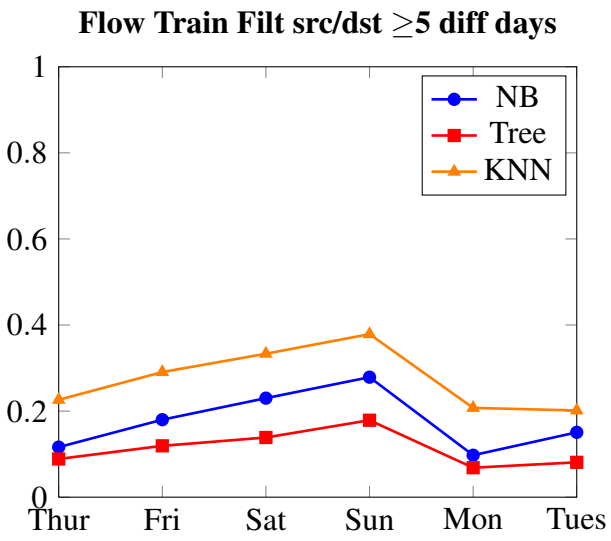

Figure 4.53: Day-to-day Accuracy (May 8-14) filtering training set for src/dst pairs that were seen at least 5 different days during the week.

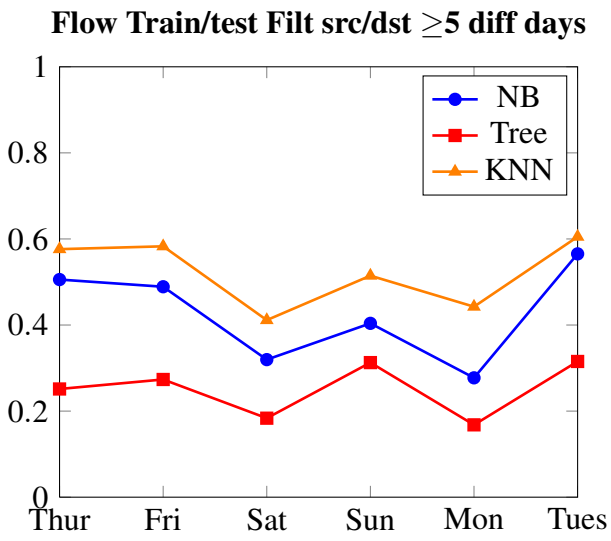

Figure 4.54: Day-to-day Accuracy (May 8-14) filtering training/test sets for src/dst pairs that were seen at least 5 different days during the week.

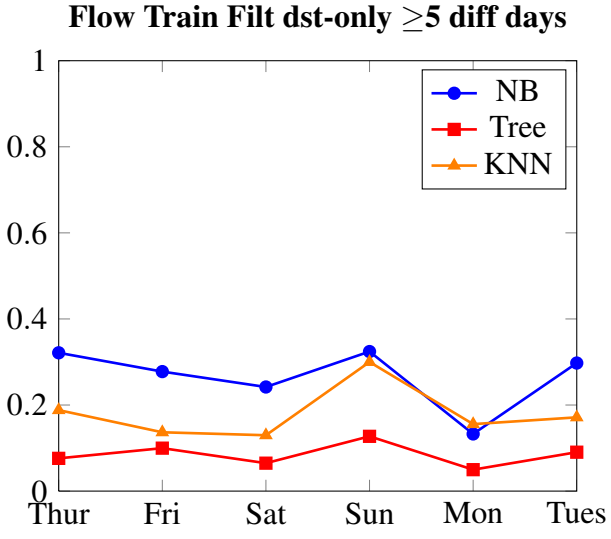

Figure 4.55: Day-to-day Accuracy (May 8-14) filtering training set for dst IPs that were seen at least 5 different days during the week.

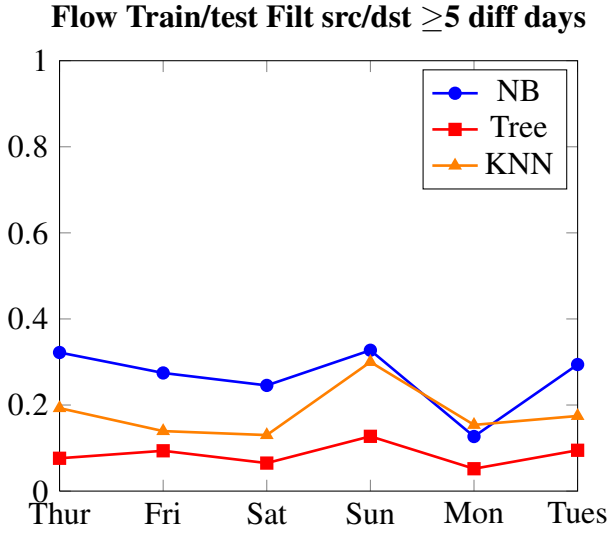

Figure 4.56: Day-to-day Accuracy (May 8-14) filtering training/test sets for dst-only IPs that were seen at least 5 different days during the week.

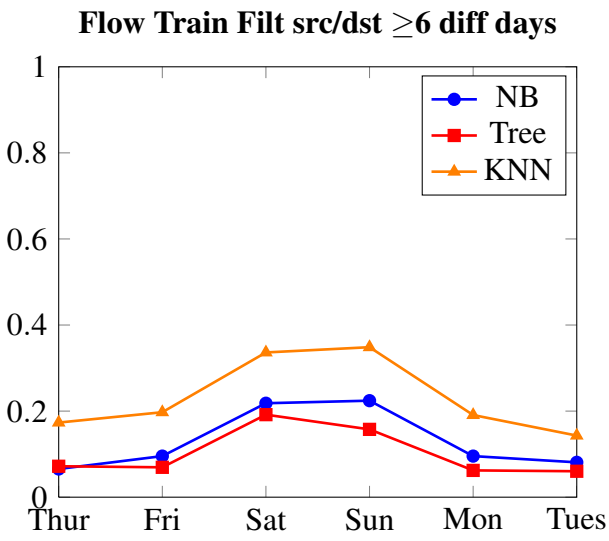

Figure 4.57: Day-to-day Accuracy (May 8-14) filtering training set for src/dst pairs that were seen at least 6 different days during the week.

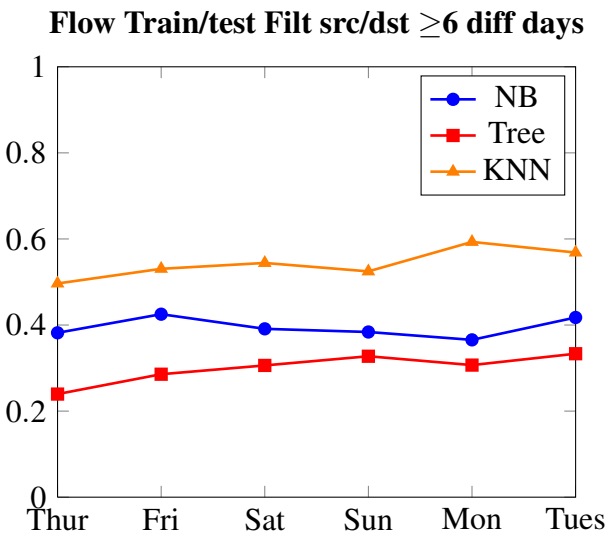

Figure 4.58: Day-to-day Accuracy (May 8-14) filtering training/test sets for dst-only IPs that were seen at least 6 different days during the week.

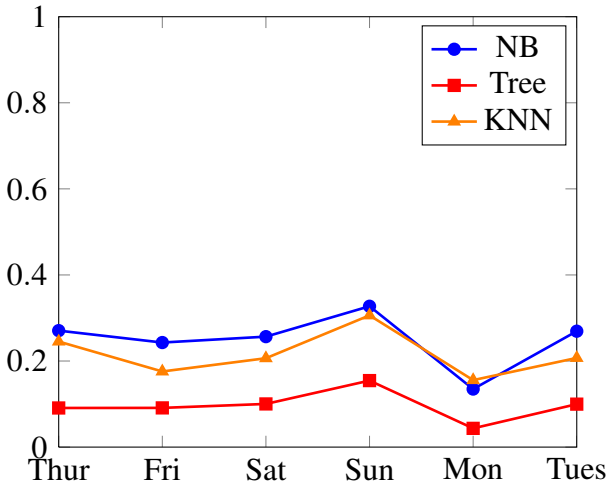

Figure 4.59: Day-to-day Accuracy (May 8-14) filtering training set for dst-only IPs that were seen at least 6 different days during the week.

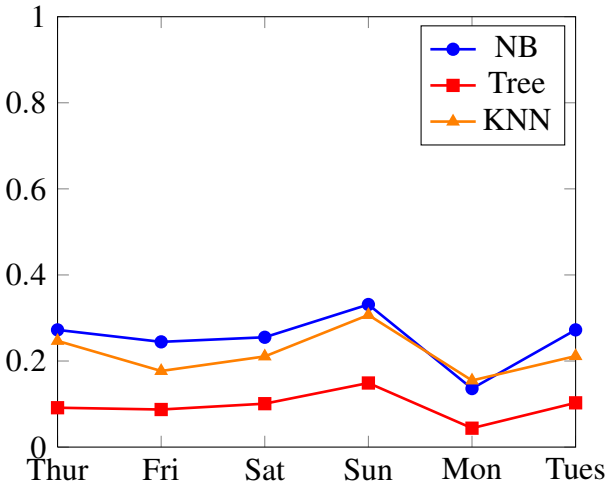

Figure 4.60: Day-to-day Accuracy (May 8-14) filtering training/test sets for dst-only IPs that were seen at least 6 different days during the week.

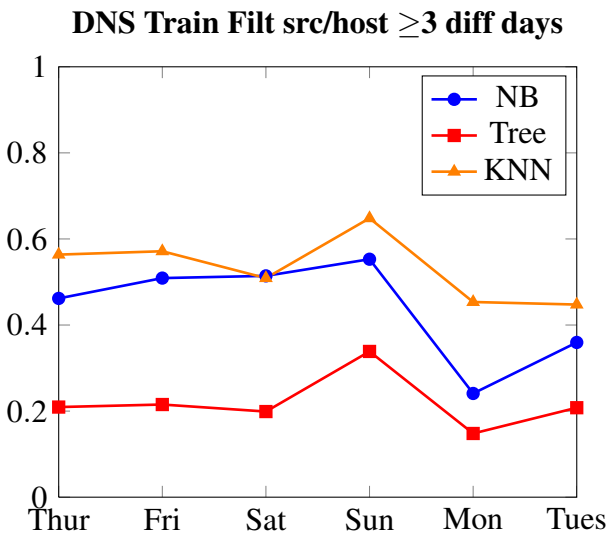

Figure 4.61: Day-to-day Accuracy (May 8-14) filtering training set for src/host pairs that were seen at least 3 different days during the week.

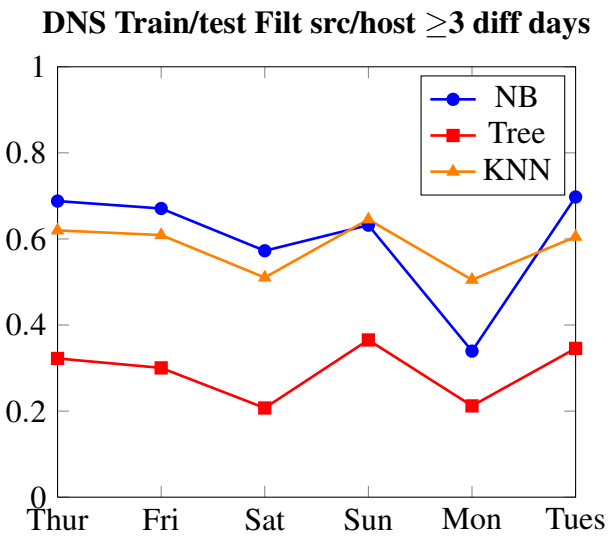

Figure 4.62: Day-to-day Accuracy (May 8-14) filtering training/test sets for src/host pairs that were seen at least 3 different days during the week.

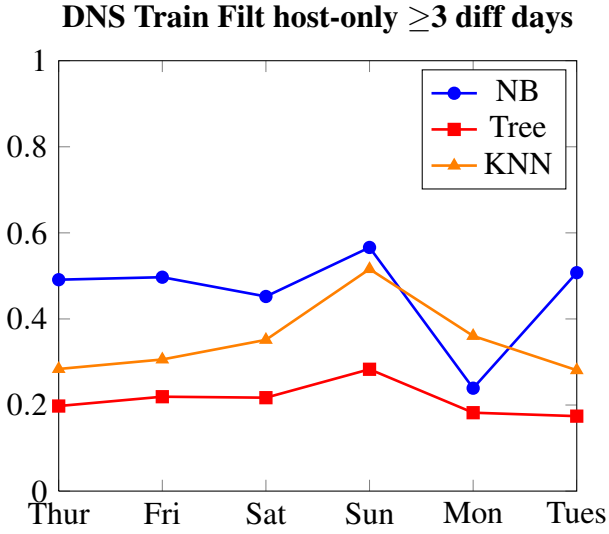

Figure 4.63: Day-to-day Accuracy (May 8-14) filtering training set for hosts-only that were seen at least 3 different days during the week.

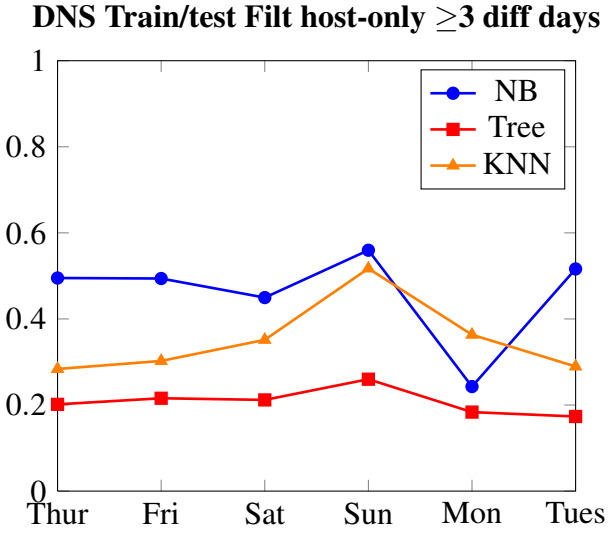

Figure 4.64: Day-to-day Accuracy (May 8-14) filtering training/test sets for hosts-only that were seen at least 3 different days during the week.

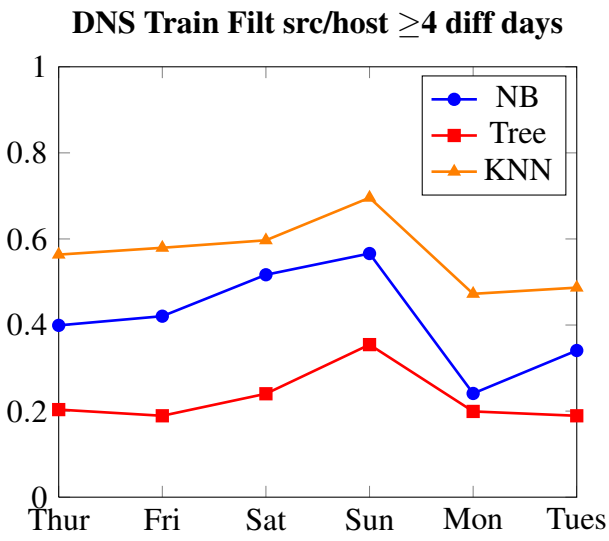

Figure 4.65: Day-to-day Accuracy (May 8-14) filtering training set for src/host pairs that were seen at least 4 different days during the week.

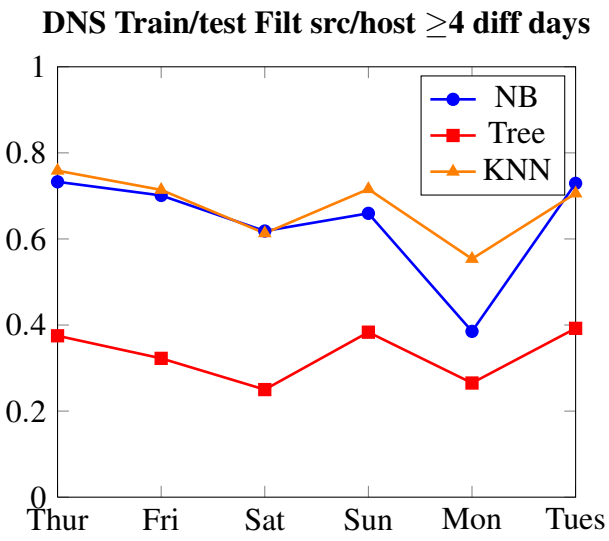

Figure 4.66: Day-to-day Accuracy (May 8-14) filtering training/test sets for src/host pairs that were seen at least 4 different days during the week.

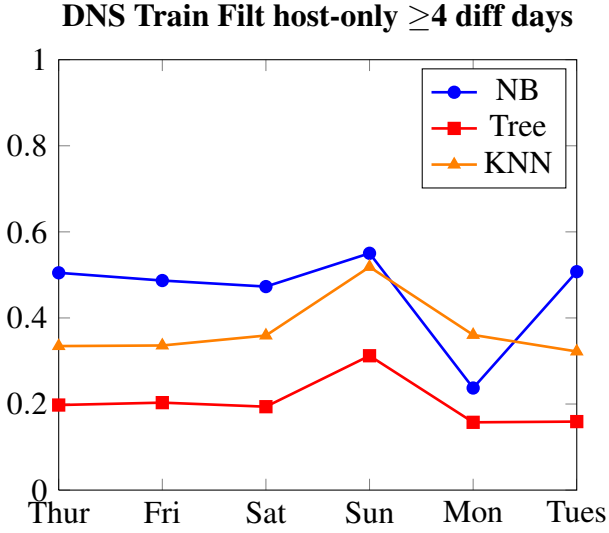

Figure 4.67: Day-to-day Accuracy (May 8-14) filtering training set for hosts-only that were seen at least 4 different days during the week.

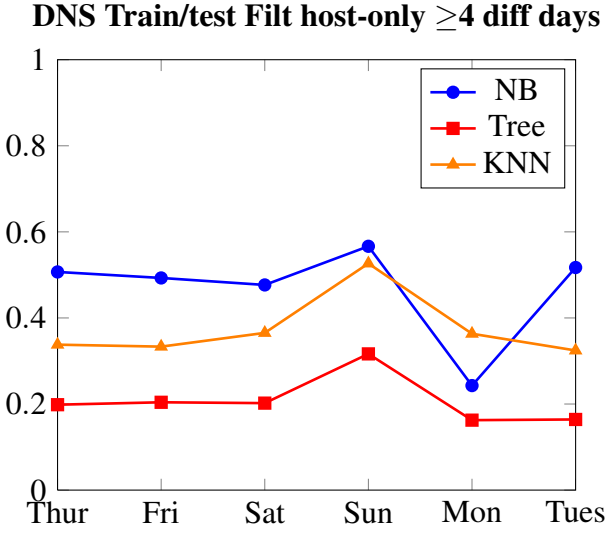

Figure 4.68: Day-to-day Accuracy (May 8-14) filtering training/test sets for hosts-only that were seen at least 4 different days during the week.

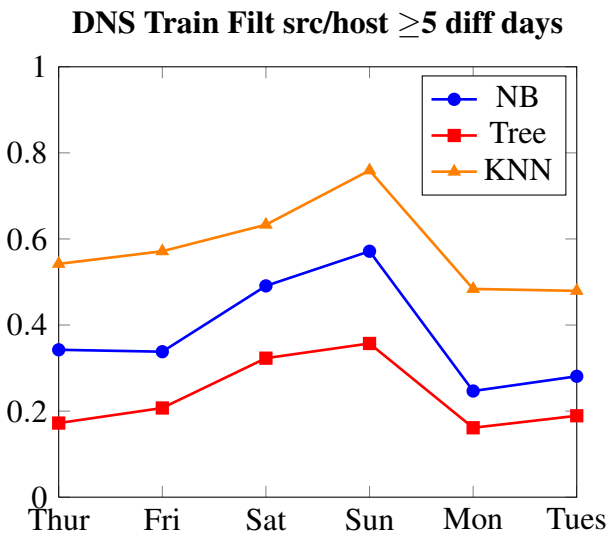

Figure 4.69: Day-to-day Accuracy (May 8-14) filtering training set for src/host pairs that were seen at least 5 different days during the week.

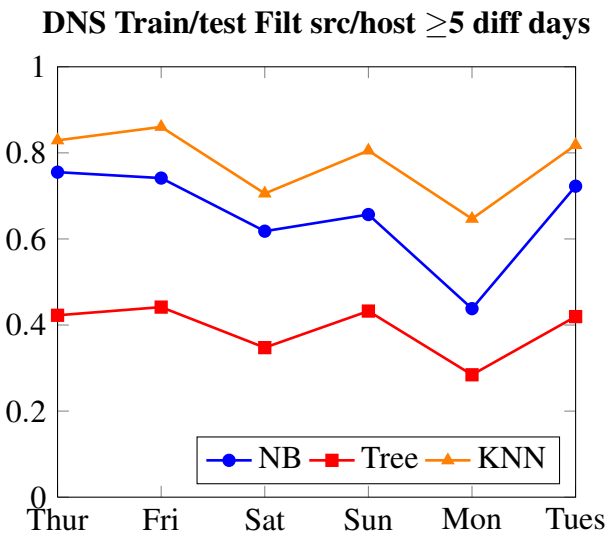

Figure 4.70: Day-to-day Accuracy (May 8-14) filtering training/test sets for hosts-only that were seen at least 5 different days during the week.

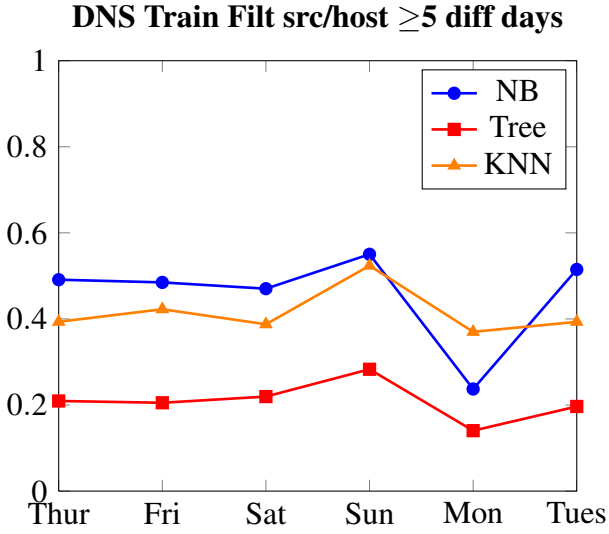

Figure 4.71: Day-to-day Accuracy (May 8-14) filtering training set for hosts-only that were seen at least 5 different days during the week.

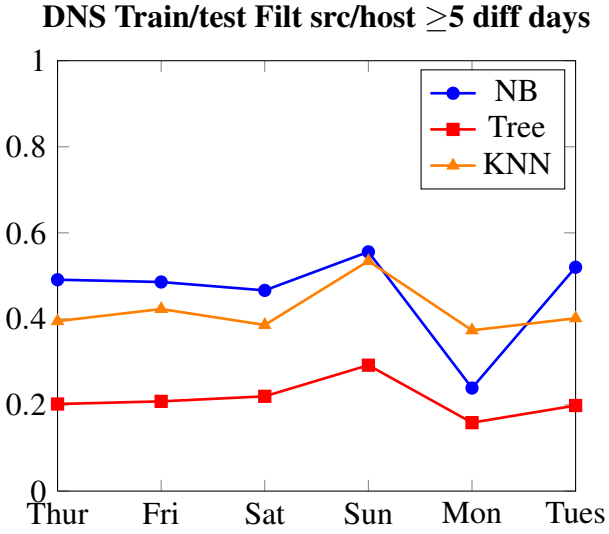

Figure 4.72: Day-to-day Accuracy (May 8-14) filtering training/test sets for hosts-only that were seen at least 5 different days during the week.

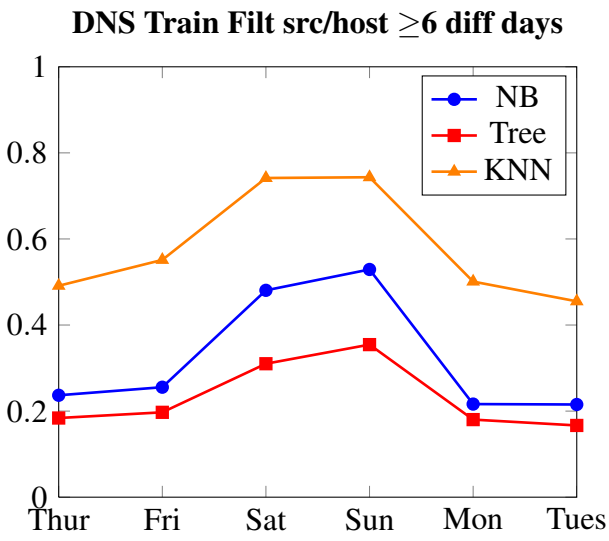

Figure 4.73: Day-to-day Accuracy (May 8-14) filtering training set for src/host pairs that were seen at least 6 different days during the week.

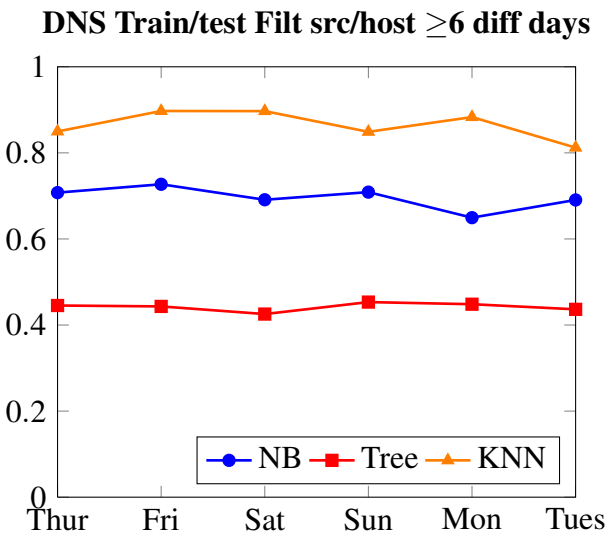

Figure 4.74: Day-to-day Accuracy (May 8-14) filtering training/test sets for hosts-only that were seen at least 6 different days during the week.

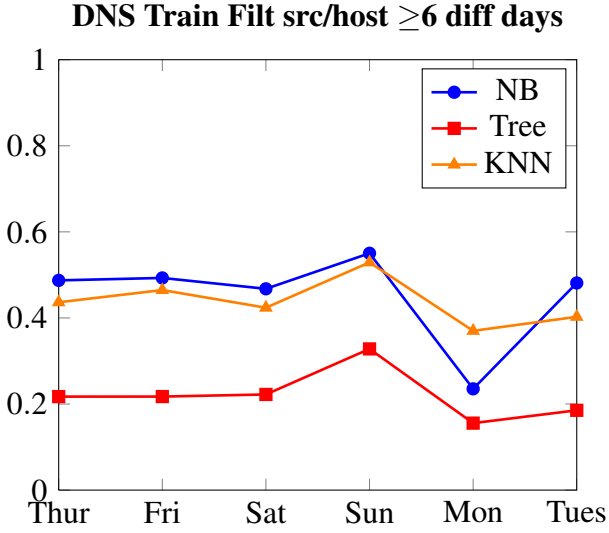

Figure 4.75: Day-to-day Accuracy (May 8-14) filtering training set for hosts-only that were seen at least 6 different days during the week.

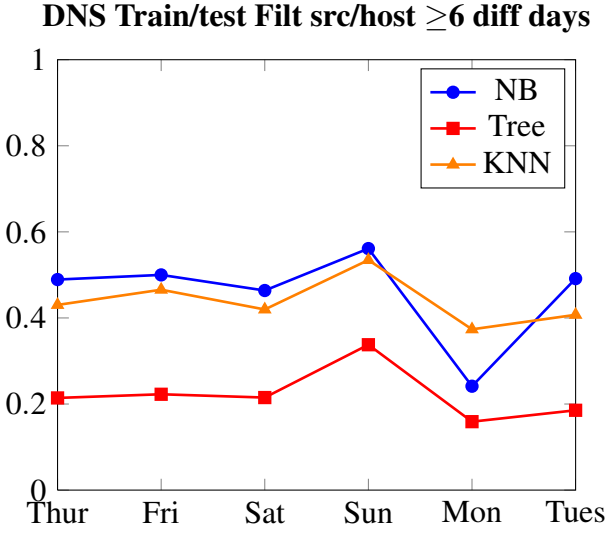

Figure 4.76: Day-to-day Accuracy (May 8-14) filtering training/test sets for hosts-only that were seen at least 6 different days during the week.

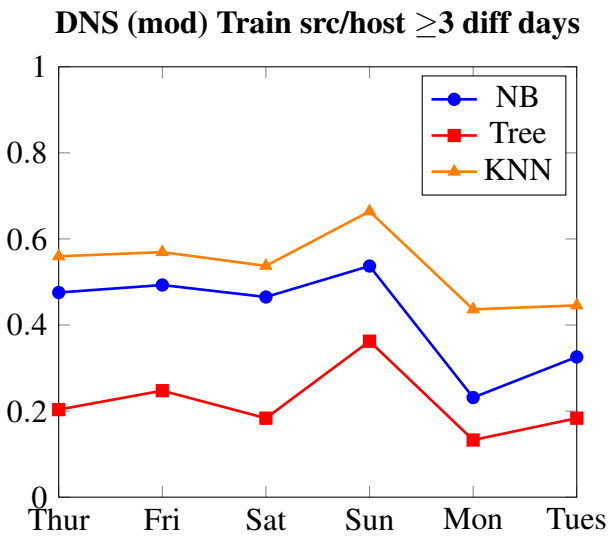

Figure 4.77: Day-to-day Accuracy (May 8-14) filtering modified training set for src/host pairs that were seen at least 3 different days during the week.

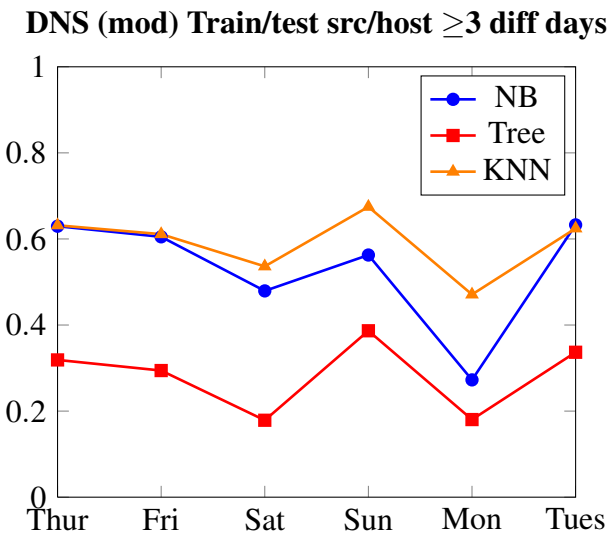

Figure 4.78: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for src/host pairs that were seen at least 3 different days during the week.

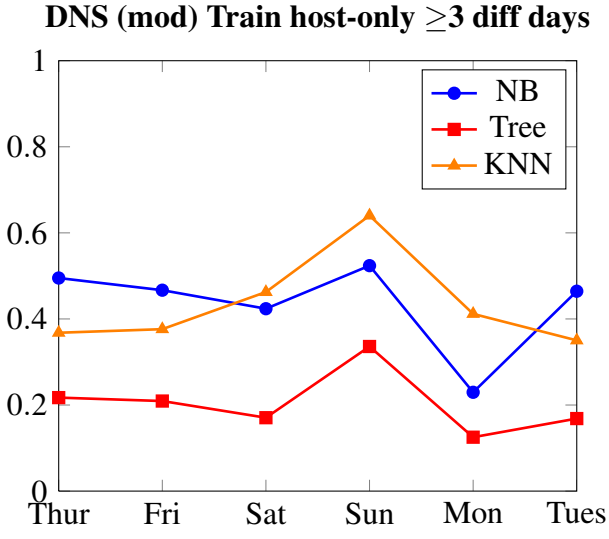

Figure 4.79: Day-to-day Accuracy (May 8-14) filtering modified training set for hosts-only that were seen at least 3 different days during the week.

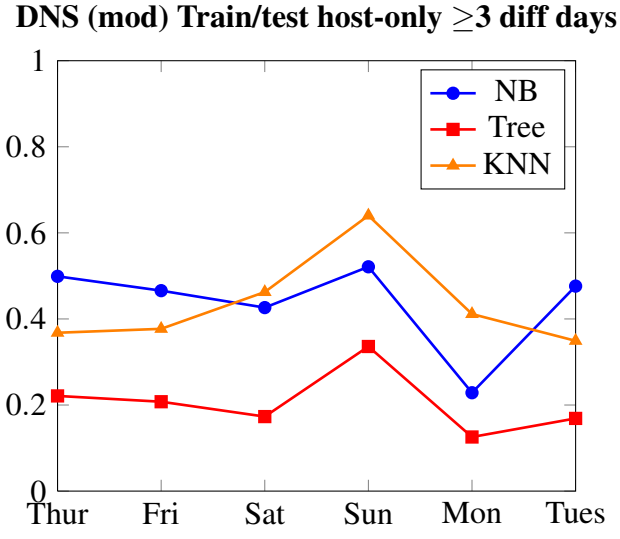

Figure 4.80: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for hosts-only that were seen at least 3 different days during the week.

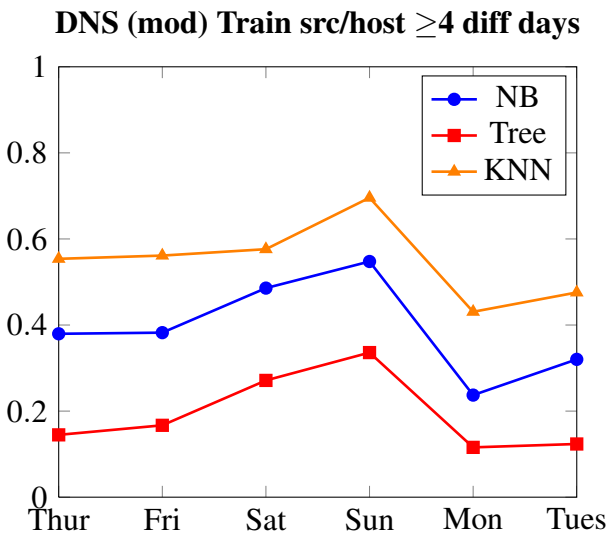

Figure 4.81: Day-to-day Accuracy (May 8-14) filtering modified training set for src/host pairs that were seen at least 4 different days during the week.

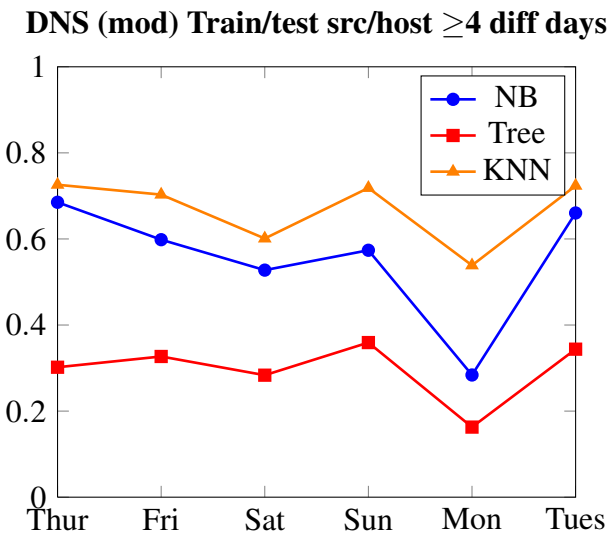

Figure 4.82: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for src/host pairs that were seen at least 4 different days during the week.

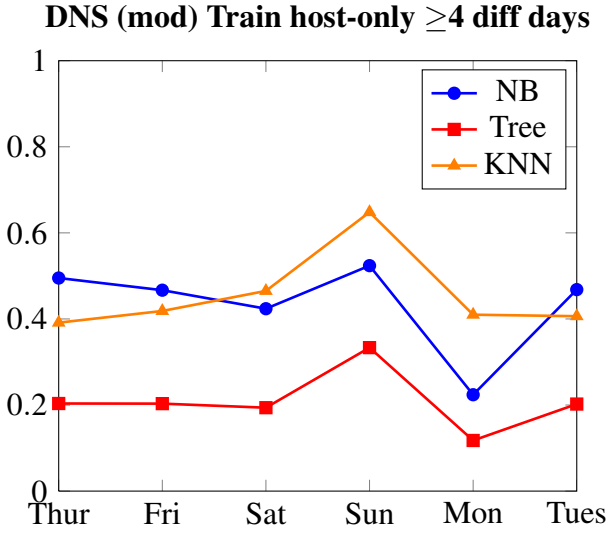

Figure 4.83: Day-to-day Accuracy (May 8-14) filtering modified training set for hosts-only that were seen at least 4 different days during the week.

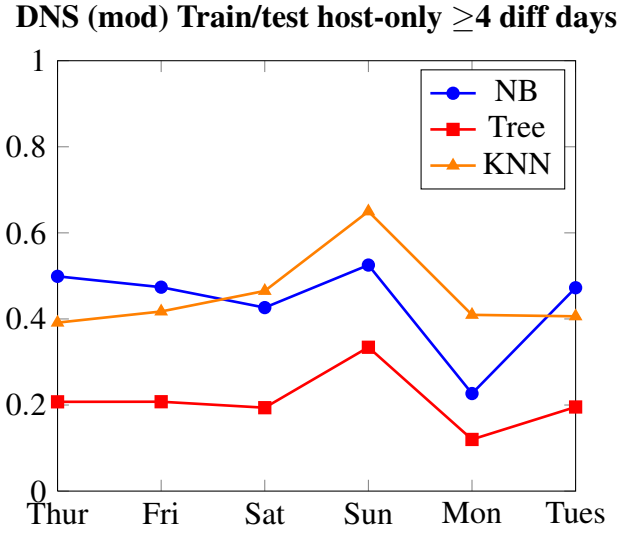

Figure 4.84: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for hosts-only that were seen at least 4 different days during the week.

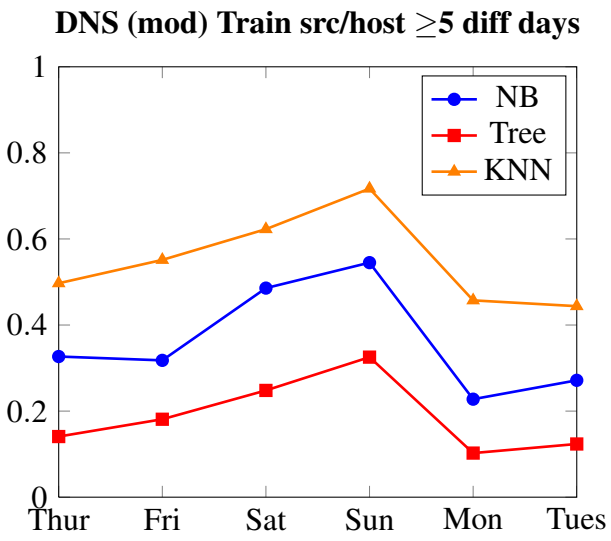

Figure 4.85: Day-to-day Accuracy (May 8-14) filtering modified training set for src/host pairs that were seen at least 5 different days during the week.

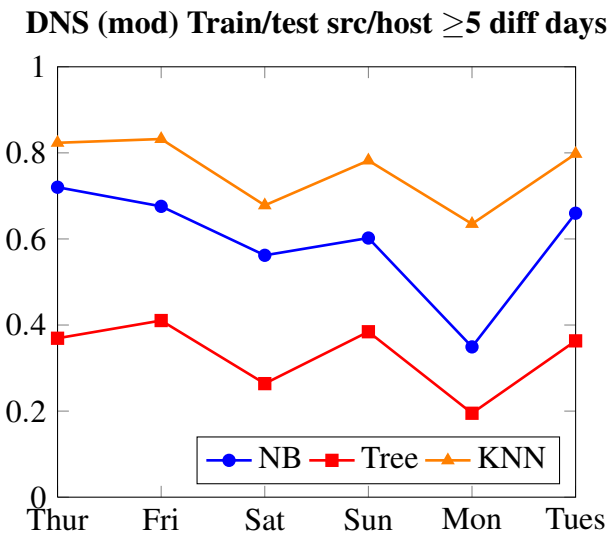

Figure 4.86: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for src/host pairs that were seen at least 5 different days during the week.

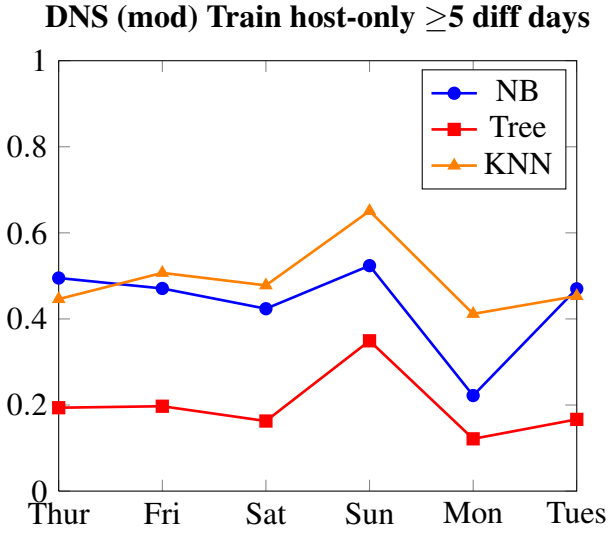

Figure 4.87: Day-to-day Accuracy (May 8-14) filtering modified training set for hosts-only that were seen at least 5 different days during the week.

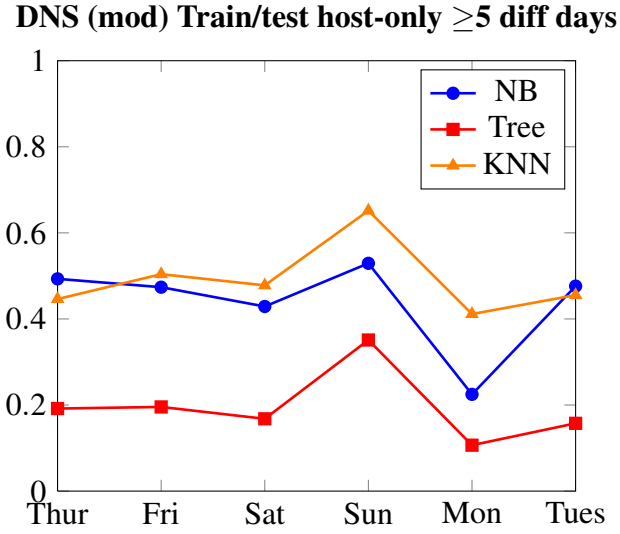

Figure 4.88: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for hosts-only that were seen at least 5 different days during the week.

DNS (mod) Train Filt src/host ≥6 diff days

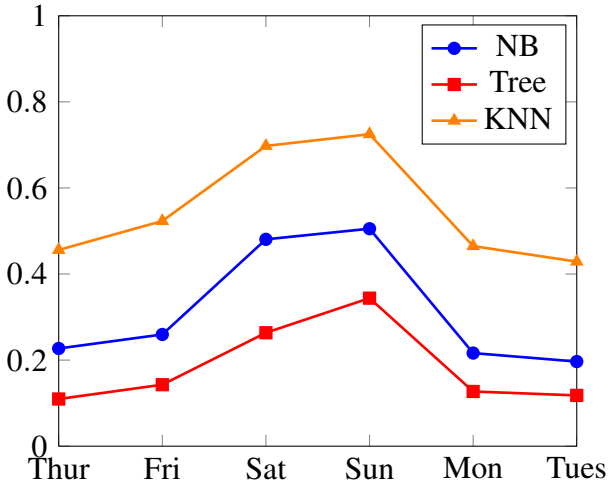

Figure 4.89: Day-to-day Accuracy (May 8-14) filtering modified training set for src/host pairs that were seen at least 6 different days during the week.

DNS (mod) Train/test Filt src/host ≥6 diff days

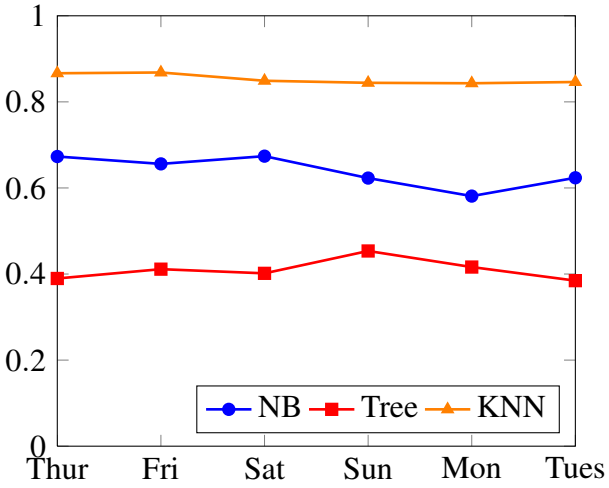

Figure 4.90: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for src/host pairs that were seen at least 6 different days during the week.

DNS (mod) Train Filt host-only ≥6 diff days

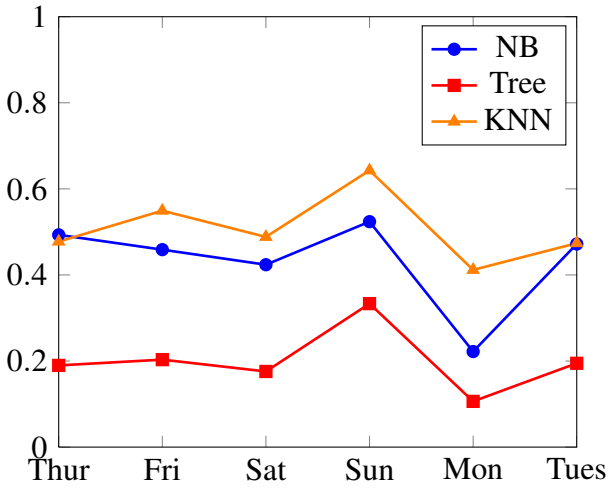

Figure 4.91: Day-to-day Accuracy (May 8-14) filtering modified training set for hosts-only that were seen at least 6 different days during the week.

DNS (mod) Train/test Filt host-only ≥6 diff days

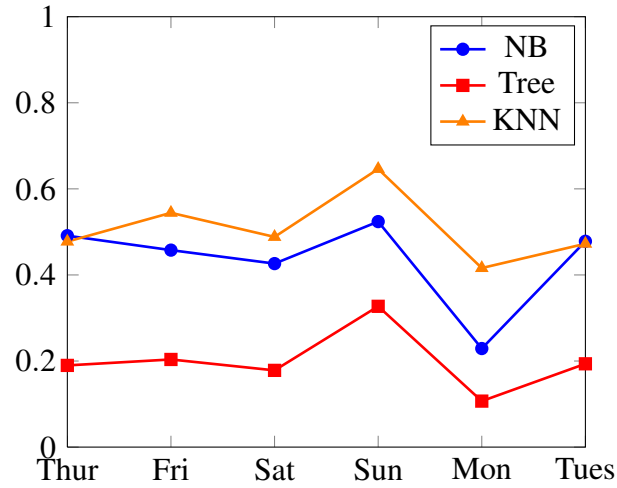

Figure 4.92: Day-to-day Accuracy (May 8-14) filtering modified training/test sets for hosts-only that were seen at least 6 different days during the week.

4.5 Orange: MDI

In this analysis, we created the training files by using multiple daily instances. For each day, all accesses within that day were aggregated, as in the "train on one day, test on the next day"

Unfiltered MDI								
Data	*# of daily instances*	*Train: # of srcs*	*dsts*	*Test: # of srcs*	*dsts*	*NB*	*Tree*	*KNN*
flow	2	531	16,438	461	10,477	3.2	21.9	11.5
	3	545	19,714	339	3,361	0.6	23.9	14.2
	4	547	20,162	330	2,584	0.3	40.0	30.0
	5	548	20,358	482	11,774	0.4	27.8	17.8
	6	573	23,697	531	11,899	1.5	27.1	17.1
DNS	2	560	55,366	497	29,885	3.2	40.6	27.2
	3	565	67,119	387	6,297	1.0	48.8	39.0
	4	566	68,205	378	4,886	0.5	76.5	54.0
	5	566	68,946	527	40,929	0.4	53.3	37.8
	6	582	86,903	534	40,900	1.3	55.4	35.8
DNS (mod)	2	560	24,616	497	12,515	3.2	42.5	35.2
	3	565	30,196	387	2,244	0.5	54.8	51.4
	4	566	30,629	378	1,840	0.0	75.9	66.7
	5	566	30,959	527	19,487	0.0	52.0	46.9
	6	582	41,107	534	19,263	1.3	55.4	45.5

Table 4.9: Accuracy (percent) using Multiple Daily Instances for the training sets (May 8-14). The x-axis indicates the test day; the number in parentheses is the number of daily instances (ranging from two when testing on Friday, to six when testing on Tuesday.

method. Each aggregated day was then added to subsequent, adjacent days to create a training file with multiple daily instances. The effect is similar to concatenating multiple tab files, each containing daily instances, to create a larger training file. The test set remained as a tab file of daily instances immediately following the final day in the training set, as in §4.4 and §4.4.4.

4.5.1 MDI: Unfiltered

When running Orange on the unfiltered unfiltered Multiple Daily Instances, the shapes of the accuracy graphs for Tree and KNN were reminiscent of using the single daily instances, with a sharp increase on Sunday, followed by a decrease on Monday. However, the NB accuracy was terrible for all analyses, hovering near the x-axis. Under the single daily instances method, NB provided the highest accuracy for all unfiltered analyses. With all the unfiltered analyses, the highest accuracy values occurred when testing on Sunday, with four daily instances in the training set, and then fell on the following day. This is likely due to fewer training instances being present in Sunday's test set, making identification easier (see Table 4.9).

Flow Data. Under the MDI method with a two-day training set, the NB was only 3.3%, which was the highest accuracy for the unfiltered MDI flow accuracy. The NB accuracy decreased with additional day added to the training set, reaching a low of 0.3% when testing on Sunday with a four-day training set, before recovering slightly on the final testing day to 1.5% (Figure 4.93). With the previous method using single daily instances, the unfiltered NB accuracy achieved a mean value of 27.1% for the week.

Another difference was that the Tree classifier recorded the highest accuracy for each test, with a low of 21.9% for a two-day training set, and a high of 40.0% with a training set composed of four days. Using unfiltered single daily instances, Tree yielded a mean of 8.5% for the week. KNN likewise improved over single daily instances, though not by as large a margin as Tree. The lowest accuracy for KNN was 13.2% on Friday, and a maximum of 30.0% on Sunday, compared to a mean of 13.5% for unfiltered single daily instances.

DNS Data. Using DNS queries in the datasets, the Tree classifier returned the highest accuracies, with a maximum value of 76.8% with four daily instances in the training set, and a minimum of 40.6% for two daily instances. (Figure 4.94). The results for KNN mimicked Tree in terms of the shape of the graph, but was lower for each test. KNN returned a maximum of 54.0% and a minimum of 27.2%. As was seen with flow data, the NB accuracy again yielded extremely poor results, with a high of 4.8%. When using unfiltered single daily instances, NB yielded a mean accuracy of 46.5%, Tree had 20.3%, and KNN returned a mean of 30.9%.

DNS (mod) Data. As with the flow data, the Tree classifier performed the best, with a minimum accuracy of 42.5% for a two-day training set, and a max of 75.9% on Sunday, (Figure 4.95), which was considerably higher than the unfiltered single daily instances, when Tree had a mean accuracy of 20.7%. NB again performed extremely poorly, with a maximum value of 3.2% with a two-day training set. Like the Tree classifier, KNN likewise improved on the mean accuracy for single daily instance method, with an MDI maximum of 66.7% and a low of 35.2%, compared to a mean of 40.0% for unfiltered single daily instances.

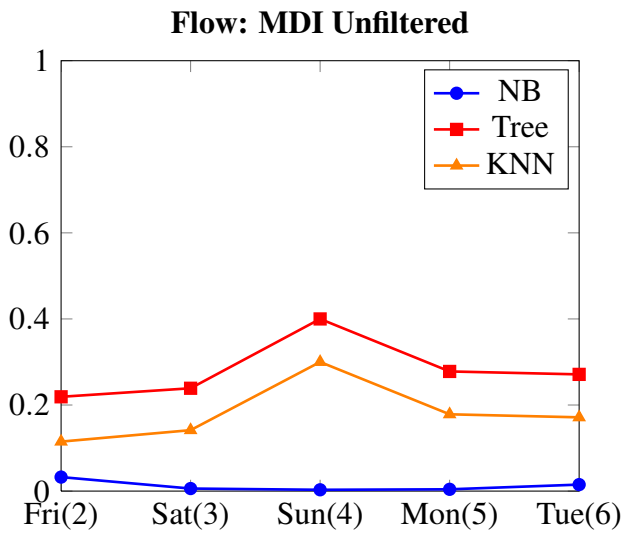

Figure 4.93: Flow Accuracy using unfiltered MDI; all days (May 8-14) prior to the test day are included in the training set. Numbers of daily instances are indicated in parentheses.

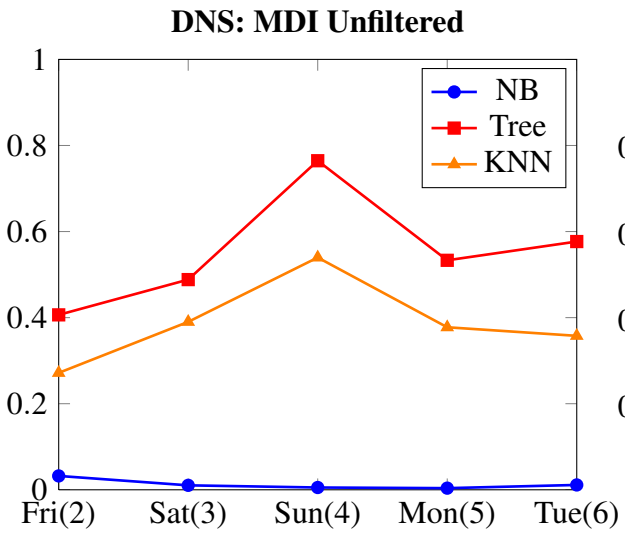

Figure 4.94: DNS Accuracy using unfiltered MDI; all days (May 8-14) prior to the test day are included in the training set. Number of daily instances are indicated in parentheses.

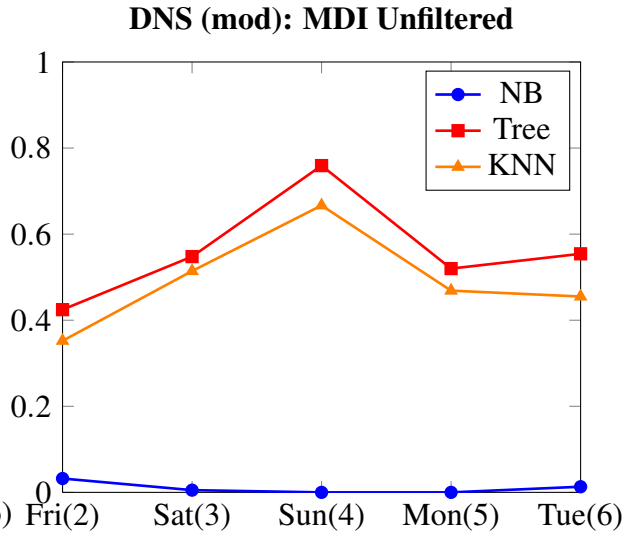

Figure 4.95: DNS (modified hostnames) Accuracy using unfiltered MDI; all days (May 8-14) prior to the test day are included in the training set. Number of daily instances are indicated in parentheses.

Flow MDI: lifetime ≥ six days									
# of daily instances	*type of filter*	*Train: # of srcs*	*dsts*	*Test: # of srcs*	*dsts*	*Set(s)* Filtered	*NB*	*Tree*	*KNN*
2	src/dst	416	3,054	461	10,477	train	1.5	13.9	13.9
				386	2,294	train+test	3.9	30.1	27.8
	dst-only	528	6,278	461	10,477	train	3.2	24.1	14.5
				459	5,131	train+test	3.2	23.1	14.6
3	src/dst	433	3,295	339	3,361	train	0.3	20.6	21.5
				297	1,275	train+test	1.3	32.3	32.3
	dst-only	541	6,569	339	3,361	train	5.9	27.7	15.6
				337	2,507	train+test	0.6	27.3	15.7
4	src/dst	435	3,334	330	2,584	train	0.0	26.7	26.7
				286	1,213	train+test	0.7	45.5	45.5
	dst-only	543	6,623	330	2,584	train	0.3	40.3	32.4
				329	2,082	train+test	0.3	40.4	32.5
5	src/dst	436	3,342	482	11,774	train	0.6	14.9	19.1
				406	2,374	train+test	3.7	47.5	49.0
	dst-only	544	6,633	482	11,774	train	0.4	24.9	20.5
				480	5,523	train+test	0.4	24.6	20.8
6	src/dst	444	3,383	531	11,899	train	0.8	19.4	36.7
				423	3,217	train+test	1.9	53.7	71.9
	dst-only	568	6,649	531	11,899	train	1.5	26.0	23.7
				528	6,427	train+test	1.5	25.9	24.2

Table 4.10: Flow Accuracy (percent) using Multiple Daily Instances filtering for flows that have sliding-window lifetimes of six days or more (May 8-14).

4.5.2 MDI: Lifetime > 6 days

For this analysis, the training and test sets were filtered to retain only src/dst or src/hostname pairs that had sliding-window lifetimes longer than six days. Based on the results using single daily instances, we only present graphs for filtering src/dst or src/hostname pairs on both training and test sets, as they yielded the highest accuracy gains.

Flow Data. As with the unfiltered MDI for flow data, the NB classifier performed rather poorly, with a high of 3.9% and a low of 0.7%. These numbers are only slightly higher than unfiltered flow data. The Tree and KNN classifiers, however, displayed steady improvement as the number of instances in the training set increased. These improvements were exhibited even on Monday, when the Tree accuracy for DNS and modified DNS decreased slightly. The Tree classifier showed a minimum accuracy of 30.1% when the training set had two daily instances. The

DNS MDI: lifetime ≥ six days									
# of daily instances	*type of filter*	*Train: # of srcs*	*dsts*	*Test: # of srcs*	*dsts*	*Set(s)* Filtered	*NB*	*Tree*	*KNN*
2	src/dst	515	10,159	497	29,885	train	1.6	23.1	35.6
				457	5,174	train+test	5.2	56.7	59.5
	dst-only	559	16,475	497	29,885	train	3.6	46.3	30.8
				495	12,849	train+test	3.6	46.5	31.1
3	src/dst	523	11,123	387	6,297	train	0.7	35.1	45.0
				360	2,237	train+test	1.1	55.3	58.6
	dst-only	563	17,325	387	6,297	train	1.0	52.7	39.0
				386	4,401	train+test	1.0	53.4	38.9
4	src/dst	523	11,224	378	4,886	train	0.5	46.6	56.9
				352	2,095	train+test	0.6	82.4	71.9
	dst-only	564	17,354	378	4,886	train	0.5	72.2	52.9
				376	3,517	train+test	0.5	72.9	52.4
5	src/dst	523	11,237	527	40,929	train	0.4	32.6	45.2
				474	5,433	train+test	2.5	73.4	75.3
	dst-only	564	17,363	527	40,929	train	0.3	51.8	38.1
				524	13,956	train+test	0.4	52.3	37.8
6	src/dst	527	11,314	534	40,900	train	0.4	38.6	62.5
				485	7,129	train+test	0.8	79.2	90.1
	dst-only	578	17,406	534	40,900	train	1.3	57.5	44.9
				525	16,603	train+test	1.3	57.7	45.5

Table 4.11: DNS Accuracy (percent) using MDI filtering for DNS queries that have sliding-window lifetimes of six days or more (May 8-14).

maximum accuracy for Tree was seen on Tuesday at 53.7%, which is 13.7% higher than the max with unfiltered MDI data, and 36.2% higher than single daily instances that were filtered for six-day lifetimes. For KNN, the high and low accuracies were 27.7% on Friday and 71.9% on Tuesday, the latter value being 41.9% higher than unfiltered MDI and 37.5% higher than single daily instances filtered for six-day lifetimes (see Table 4.10 and Figure 4.96).

DNS Data. When using DNS query data, NB again was quite low, with a max of 2.5%. The Tree and KNN classifiers performed better, and displayed similar behavior with each other as the numbers of instances in the training sets increased. Interestingly, the lowest accuracy for both Tree and KNN occurred on Saturday, with three instances in the training set. Unlike the flow data, which increased with every added daily instance, the Tree classifier had a maximum accuracy of 82.4% on Sunday, with four daily instances in the training set, although it recovered

DNS (mods) MDI: lifetime $\geq$ six days									
# of daily instances	*type of filter*	*Train: # of srcs*	*dsts*	*Test: # of srcs*	*dsts*	*Set(s)* Filtered	*NB*	*Tree*	*KNN*
2	src/dst	495	2692	497	12,515	train	2.2	20.9	35.8
				461	1,905	train+test	4.8	56.8	60.7
	dst-only	560	6381	497	12,515	train	3.4	38.2	38.0
				496	4,867	train+test	3.4	39.3	38.3
3	src/dst	501	2903	387	2,244	train	0.3	33.1	43.9
				360	735	train+test	0.3	51.1	63.6
	dst-only	564	6812	387	2,244	train	0.5	46.3	49.4
				387	1,478	train+test	0.5	46.5	49.4
4	src/dst	501	2916	378	1,840	train	0.0	46.0	56.3
				354	712	train+test	0.0	79.9	78.8
	dst-only	565	6823	378	1,840	train	0.0	73.3	64.8
				377	1,478	train+test	0.0	73.2	65.0
5	src/dst	501	2918	527	19,487	train	0.4	30.2	44.2
				477	2,031	train+test	2.1	68.1	75.7
	dst-only	565	6826	527	19,487	train	0.0	50.7	44.6
				525	5,361	train+test	0.0	50.9	45.1
6	src/dst	505	2937	534	19,263	train	0.4	30.9	59.2
				487	2,805	train+test	1.0	78.4	87.9
	dst-only	579	6842	534	19,263	train	1.3	53.7	49.8
				528	6,437	train+test	1.3	54.4	50.9

Table 4.12: DNS (mod) Accuracy (percent) using MDI filtering for modified DNS hostnames having sliding-window lifetimes of six days or more (May 8-14).

to 79.2% when using six daily instances. The minimum for tree was on Saturday, at 55.3%. With the exception of a slight decrease of 0.9% from Friday to Saturday, KNN, generally continued to improve with each additional instance added to the training set, reaching a maximum accuracy of 90.1% on Tuesday. This value is 54.3% higher than when using unfiltered MDI, and 30.6% better than the six-day filter for single daily instances. (see Table 4.11 and Figure 4.97).

DNS (mod) Data. The results using modified DNS hostnames were generally less accurate than unmodified DNS data when using the Tree classifier, and more accurate with KNN. As with the unmodified DNS data, the Tree classifier again hit a minimum accuracy on Saturday, this time at 51.5% before hitting 79.9% the following day. KNN increased with each successive addition of training instances, except for a small dip on Monday, followed by the maximum accuracy on Tuesday of 87.9% (see Table 4.12 and Figure 4.98).

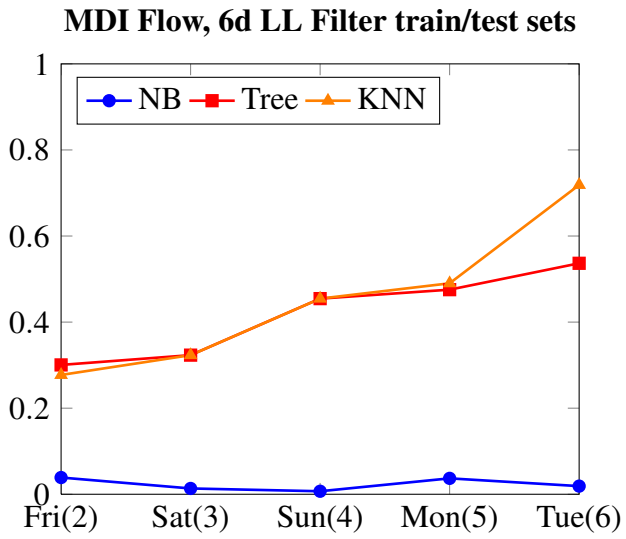

Figure 4.96: MDI Flow Accuracy retaining only long-lived src/host pairs in both the training and test sets (May 8-14). Numbers of daily instances are indicated in parentheses.

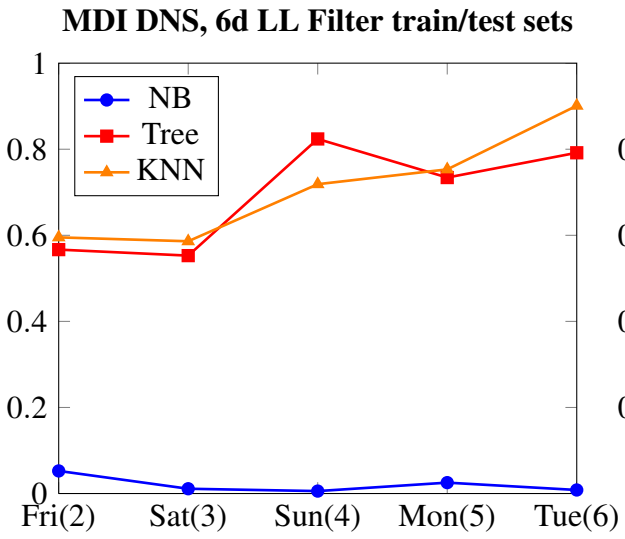

Figure 4.97: MDI DNS Accuracy retaining only long-lived src/dst pairs in both the training and test sets (May 8-14). Numbers of daily instances are indicated in parentheses.

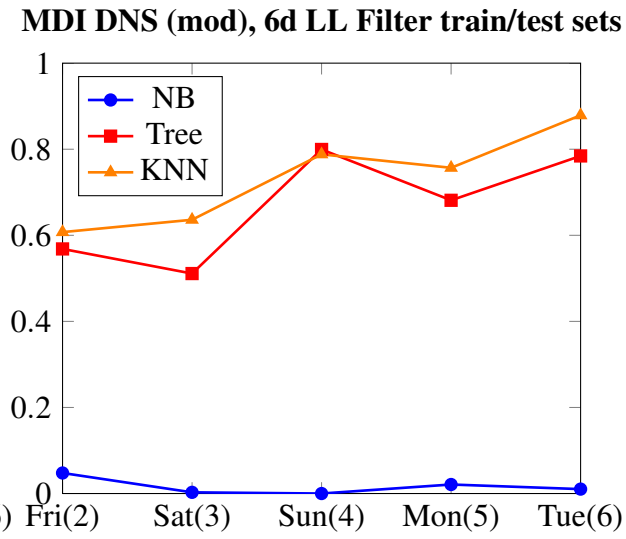

Figure 4.98: MDI DNS (mod) Accuracy retaining only long-lived src/host pairs in both the training and test sets (May 8-14). Numbers of daily instances are indicated in parentheses.

4.5.3 Implications

The NB results for MDI were extremely low, even when only retaining long-lived accesses and DNS queries. This is in sharp contrast to the results of single daily instances, wherein NB accuracy performed rather well, and displayed great increases in accuracy when only retaining the LL flows.

For training sets composed of two, three, and four days, the Tree and KNN accuracy increase, which seems to suggest that the extra information adds to the classifiers' ability to correctly predict a class, at least in the short term. The significant drop with a five-day unfiltered training set might suggest that the behavior of users in our dataset is sufficiently different from one week to the next that large training sets will incorporate large amounts of irrelevant data that serves as noise. The amount of noise would then increase with the number of days in the training set. It could also reflect the same feature as the single daily instances, where fewer users on the weekend led to higher accuracies for those days. As seen in Table 4.9, the numbers of sources in the test sets on Saturday and Sunday are considerably lower than on weekdays. To that end, it is likely that the users on the weekend were present on some weekdays, so having trained on a larger set made identification of these few users easier. That said, it can also be seen that Tree and KNN accuracies were higher on Monday (with a five-day training set) than on the previous Friday, which had a two-day training set; with the single daily instances, the accuracy on Monday was significantly lower than on any other day. Thus, even with the unfiltered MDI, it would appear that more instances in the training sets serves to increase accuracy.

This relationship seems to be supported by the analysis using the filtered training and test sets (Tables 4.10, 4.11, and 4.12). The accuracy Monday is still lower than on Sunday, but is significantly higher that other previous days, and continues to increase with each added day in the training set.

4.6 Associated ASNs for Flow Data

For each unique source/destination IP and unique destination IP (irrespective of source) observed in May 8-14, we looked up the Autonomous System Number (ASN) announcing the most specific BGP prefix corresponding to the destination IP. We then determined which ASNs were the most popular in our dataset when considering just the short-lived accesses, just the long-lived accesses, and the dataset as a whole. We then obtained the organization name for the top ten most popular ASNs for each group by querying the Regional Internet Registry (RIR) whois databases.

4.6.1 Top-ten ASes for src/dst IP Pairs

There were 2,541 unique ASNs for the week of May 8-14, 2013. Table 4.13 shows the ten most commonly visited ASes when considering src/dst pairs. Unsurprisingly, Amazon, Google, Microsoft, and various CDNs are present in the list. In Table 4.14 are shown the ten most commonly visited ASes among the SL src/dst pairs. Note that here we are investigating the SL pairs themselves, vice omitting them as we did with the accuracy analyses in §4.4 and §4.5.

Comparing SL ASes to All ASes. Interestingly, the popular ASes for short-lived traffic are very similar to the popular ASes for all src/dst accesses, with only one different AS in the top-ten list. The discrepancy is that the top-ten list for all flows contains an AS for the California State University (CSU) Network (ranked #13 with 1.42% for SL flows), and the top-ten list for SL flows has an AS for Comcast (ranked #11 with 1.66% for all flows). It makes sense that the top ASes for the SL flows would be near the top, as SL flows accounted for roughly 50% of the total number of flows (§4.1.1). That the top-ten ASes for SL flows are virtually identical, with the different ASes being present just outside the top-ten list, indicates that even SL flows are destined for the overall popular ASes.

Comparing LL ASes to All ASes. Table 4.15 shows the top-ten ASes from just the long-lived flows with sliding-window lifetimes of more than six days. There are two differences with the top-ten ASes for all flows. The ninth and tenth most popular ASes for the full list are AppNexus (ranked #29 with 0.60% on the LL list) and Level 3 Communications (ranked #11 with 1.85% for LL flows). For the LL list of ASes, the third most popular AS is 0, indicating private IP space (#14 with 1.48% on the full list), and the seventh most popular long-lived AS is Edgecast Networks (#15 with 1.43% on the full list).

Comparing LL ASes to SL ASes. Comparing the ASes for long-lived flows with ASes for short-lived flows, we see three discrepancies. AppNexus, Level 3 Communications, and Comcast are present in the SL list, but ranked 29th (0.60%), 13th (1.85%), and 12th (1.76%), respectively, on the LL list. Conversely, the LL list had ASes for the CSU Network, private IP space, and EdgeCast Networks, which ranked 13th (1.42%), 14th (1.17%), and 19th (0.82%), respectively, on the SL list.

The similarity between the long-lived and short-lived lists is interesting. Nine of the SL ASes were present within the top twelve of long-lived ASes, and nine of the LL ASes were within the top 14 of the SL ASes. Unlike the full list of ASes, the SL flows are not present in the list

ASN	*Percentage*	*Organization*
14618	11.43	Amazon.com, Inc.
15169	10.19	Google Inc.
16509	10.01	Amazon.com, Inc.
8075	4.08	Microsoft Corp
2152	3.91	California State University Network
2914	3.84	Akamai Technologies, Inc. & NTT America, Inc.
20940	2.97	Akamai Technologies, Inc.
36351	2.83	SoftLayer Technologies Inc.
29990	2.65	AppNexus, Inc
3356	1.73	Level 3 Communications, Inc.

Table 4.13: Top 10 ASes for a one-week sliding window of all src/dst accesses (May 8-14).

ASN	*Percentage*	*Organization*
14618	13.41	Amazon.com, Inc.
16509	12.02	Amazon.com, Inc.
15169	7.68	Google Inc.
8075	4.05	Microsoft Corp
2914	4.04	Akamai Technologies, Inc. & NTT America, Inc.
29990	3.50	AppNexus, Inc
20940	3.10	Akamai Technologies, Inc.
36351	2.98	SoftLayer Technologies Inc.
7922	1.81	Comcast Cable Communications, Inc.
3356	1.58	Level 3 Communications, Inc.

Table 4.14: Top 10 ASes for short-lived destination IPs for a one-week sliding window of src/dst pairs during May 8-14.

of LL flows. The resemblance between the two lists indicates that most flows have destination IP addresses belonging to the same ASes, regardless of whether the dst IP is visited frequently or rarely by the user. The differences between the LL and SL AS lists are likewise interesting, albeit understandable. The CSU Network accounted for 13.42% of all long-lived flows, but only 1.42% of all SL flows. This seems to make sense, as the IPs for NPS belong to the CSU AS, and NPS websites would obviously be regularly accessed by NPS students and faculty.

4.6.2 Top-ten Common ASes for dst-only IP Addresses

Similar to §4.6.1, for each unique destination IP, irrespective of source, we looked up the Autonomous System Number and organization name. There were still 2541 unique ASNs, but the popularity rankings were slightly different.

ASN	*Percentage*	*Organization*
15169	15.51	Google Inc.
2152	13.42	California State University Network
0	6.34	private IP space
14618	4.54	Amazon.com, Inc.
8075	4.09	Microsoft Corp
16509	3.98	Amazon.com, Inc.
15133	3.26	EdgeCast Networks, Inc.
36351	2.33	SoftLayer Technologies Inc.
20940	2.18	Akamai Technologies, Inc.
2914	2.13	Akamai Technologies, Inc. & NTT America, Inc.

Table 4.15: Top 10 ASes for long-lived (more than 5 days) src/dst pairs for a one-week sliding window during May 8-14.

Comparing SL ASes to All ASes. The top-ten list for the full list of destination IPs, regardless of source, was similar to the AS list for SL flows. Of note, Google did not break the top-ten for short-lived ASes, but was #15 with 1.04%. As with the full list of ASes, Rackspace Hosting had 1.25% on the SL list of ASes, but was slightly lower in ranking at #12. Looking at the SL top-ten, GoDaddy.com was #12 (1.10%) for the full list, and CloudFlare was #11 (1.11%). As with the src/dst AS comparison, it is not surprising that the full list and the SL list would be somewhat similar, as the SL flows accounted for 40% of all flows. That the remaining 60% did not change the rankings much indicates the overall popularity of these ASes.

Comparing LL ASes to All ASes. The top-ten list of long-lived ASes had nine of the top-ten ASes for all flows. Rackspace Hosting was tenth on the full list, but 16th on the LL list, with 0.86%. The LL AS list instead had AppNexus in the tenth spot, which was #14 on the full list at 0.80%.

Comparing LL ASes to SL ASes. The comparison of long-lived ASes to short-lived ASes reveals two discrepancies. The AS for Google was the fourth most popular AS for LL flows with 5.97%, but was #16 for SL ASes (1.04%). AppNexus ranked tenth (1.49%) for LL ASes, but was #90 (0.13%) for SL ASes. Conversely, GoDaddy.com and CloudFlare, seventh (1.85%) and eighth (1.63%) on the SL list were #43 (0.28%) and #24 (0.55%) on the LL list. This could be explained by the regularity with which users tend to visit Google websites, including the popular searching and mail programs.

ASN	*Percentage*	*Organization*
14618	9.89	Amazon.com, Inc.
16509	7.66	Amazon.com, Inc.
0	4.33	private IP space
2914	4.10	Akamai Technologies, Inc. & NTT America, Inc.
15169	3.71	Google Inc.
7922	3.03	Comcast Cable Communications, Inc.
36351	2.71	SoftLayer Technologies Inc.
8075	2.15	Microsoft Corp
20940	1.93	Akamai Technologies, Inc.
33070	1.25	Rackspace Hosting

Table 4.16: Top 10 ASes for a one-week sliding window of all dstIP accesses, irrespective of source IP (May 8-14).

ASN	*Percentage*	*Organization*
16509	7.35	Amazon.com, Inc.
14618	5.55	Amazon.com, Inc.
2914	4.36	Akamai Technologies, Inc. & NTT America, Inc.
0	3.55	private IP space
7922	3.54	Comcast Cable Communications, Inc.
36351	2.96	SoftLayer Technologies Inc.
26496	1.85	GoDaddy.com, LLC
13335	1.63	CloudFlare, Inc.
20940	1.42	Akamai Technologies, Inc.
8075	1.36	Microsoft Corp

Table 4.17: Top 10 ASes for short-lived destination IPs, irrespective of source IP, for a one-week sliding window (May 8-14).

ASN	*Percentage*	*Organization*
14618	14.89	Amazon.com, Inc.
0	8.74	*private IP space*
16509	8.26	Amazon.com, Inc.
15169	5.97	Google Inc.
8075	3.49	Microsoft Corp
36351	2.96	SoftLayer Technologies Inc.
7922	2.61	Comcast Cable Communications, Inc.
20940	1.68	Akamai Technologies, Inc.
2914	1.56	Akamai Technologies, Inc. & NTT America, Inc.
29990	1.49	AppNexus, Inc

Table 4.18: Top 10 ASes for long-lived (more than 5 days) destination IPs, irrespective of source IP, for a one-week sliding window (May 8-14).

Type	*ASN*	*Organization*	*src/dst*	*dst-only*
SL	15169	Google Inc.	7.68%	1.04%
	29990	AppNexus, Inc	3.50%	0.13%
	3356	Level 3 Communications, Inc.	1.58%	0.48%
	0	private IP space	3.55%	1.17%
	26496	GoDaddy.com, LLC	0.31%	1.85%
	13335	CloudFlare, Inc.	0.59%	1.63%
LL	2152	California State University Network	13.42%	0.51%
	15133	EdgeCast Networks, Inc.	3.26%	0.69%
	7922	Comcast Cable Communications, Inc.	1.76%	2.61%
	29990	AppNexus, Inc	0.60%	1.49%
Full	2152	California State University Network	3.91%	0.28%
	29990	AppNexus, Inc	2.65%	0.80%
	3356	Level 3 Communications, Inc.	1.73%	0.65%
	0	private IP space	1.48%	4.33%
	7922	Comcast Cable Communications, Inc.	1.66%	3.03%
	33070	Rackspace Hosting	0.40%	1.25%

Table 4.19: Differences between src/dst pairs and dst-only with the short-lived, long-lived, and full Top 10 AS lists (May 8-14).

4.6.3 Comparison of src/dst and dst-only Top-ten ASes

The final comparison is between the most common Autonomous System (AS)es for src/dst pairs, and dst IPs only, irrespective of source. As seen in Table 4.19, the biggest differences in each category are Google for SL ASes, and the CSU Network for the LL and the full list of ASes. The differences here are likely due to the fact that the list of IPs used to build the AS lists are unique src/dst pairs and unique dst IPs. For example, as NPS is on the CSU Network, one would expect the CSU AS to be regularly visited by most, if not all, users. However, if NPS only has a few IPs (compared to Google or Amazon), then there would be fewer unique IPs that would point to the CSU AS. Thus, if most users visit the exact same NPS IP on a near-daily basis, it would have a high AS percentage under the src/dst method, which it did, at 13.42%. However, since (in this notional example) there was only one IP being visited, the lack of variety in IPs would lead to a low AS percentage when using the dst-only approach, which, at 0.51%, was likewise the case in our study. This same reasoning would explain why Google accounted for 15.51% of the src/dst AS, but 5.97% for the dst-only method. That 5.97% is still rather high might reflect that Google has many different IPs with which to provide its very popular services.

Flow Traffic (Unfiltered)							
Test	*Day*	*mean*	*median*	*mode*	*mode cnt*	*features*	*sources*
NB	Thu	459	242	0	4	11,381	162
	Fri	336	228	319	4	10,478	128
	Sat	124	77	2	4	3,362	86
	Sun	94	50	4	3	2,585	107
	Mon	593	305	0	2	11,775	64
	Tue	271	241	35	3	11,900	157
Tree	Thu	227	109	0	10	11,381	40
	Fri	230	83	0	8	10,478	33
	Sat	56	9	0	13	3,362	28
	Sun	98	30	0	13	2,585	46
	Mon	565	197	0	8	11,775	30
	Tue	512	79	0	33	11,900	38
KNN	Thu	137	20	0	27	11,381	59
	Fri	171	19	0	16	10,478	40
	Sat	41	9	0	11	3,362	30
	Sun	40	6	0	33	2,585	81
	Mon	234	5	0	31	11,775	65
	Tue	148	4	0	33	11,900	69

Table 4.20: Flow - mean, median, mode, mode count of the maximum number of features that can be changed and still correctly identify the source IP. Also shown are the total number of features and the number of sources that were originally correctly predicted (May 8-14).

It can also be seen from Table 4.19 that Google accounted for 7.68% of SL src/dst ASes, but only 1.04% of dst-only AS. This difference is possibly due to many users accessing an IP on the Google AS for a single network session. If those same users visited IPs belonging to many other ASes, the variety in those other SL accesses could explain why Google was not higher on the list of dst-only SL ASes.

4.7 Profile Strength

In this analysis, we tested the whether our user network behavior profiles would detect abnormal user behavior which might indicate malicious or otherwise inappropriate activity. Due to time constraints, we only tested profiles relying on destination IPs, not those profiles using DNS queries. The abnormal network behavior was simulated by modifying the fields in the tab files (§3.5) to indicate different values for the number of times each source sent a SYN packet to a particular destination IP. As the tab file headers contained columns for every destination IP visited by every source during the training or testing period, the majority of destination IPs

for a given source had observed values of zero, indicating that the source did not send any SYN packets to those IP addresses. Thus, while we were not able to inject completely unseen destination IPs into the tab file, this method was sufficient for the purpose of simulating network activity that was abnormal.

In §4.4, §4.4.4, and §4.5, we computed user identification accuracies using Orange's NB, Classification Tree, and KNN modules. Using those results, we identified the source IPs that were correctly identified by at least one of the classifiers. For each correctly identified source, we changed the value of a randomly selected feature to a random number between 0 and 30). The purpose was to simulate the situation of Bob using Alice's computer. That is, the source IP was the same, but the real user and, hence, the network behavior, was different. After changing the value of the feature, we then assessed whether Orange would return the same (previously correct) prediction for the class. If the prediction was still for Alice, we changed another randomly selected feature to a random value between 0 and 30, and had Orange make another prediction. If the prediction returned a different source (any source other than Alice), we recorded the number of features that we were able to change before the predicted source was different.

If more than 50 features can be changed and still allow Orange to correctly identify the sources, we considered those profiles to be strong, as they were able to absorb changes. If changing only fewer than 50 features caused sources to be misidentified, we regarded those profiles as being weak, since they failed with only minor deviations in behavior.

For the case of Bob using Alice's computer without her knowledge, it would be better for Bob if the profiles tended to have high (more than 50) median values, as this would imply that at least half of the profiles would still predict Alice if Bob limited his network accesses. Recall from §4.4 and §4.4.4 that as the filters became more stringent, the numbers of destinations in the datasets decreased significantly. For the datasets using IP destinations, the mean of the number of destinations visited in the unfiltered sets was 9,046 (for all sources combined). After a long-lived filter using a six-day lifetime was applied to src/dst pairs, the mean destination IPs fell to 2,213. There were only 1,721 dst IPs in the *at least 5 days* analysis, and 1,187 dst IPs in the *at least 6 days* tests. The increase in our identification accuracy came at the cost of having fewer source and destination IPs in our datasets.

Tables 4.20, 4.21, 4.22, and 4.23 show the mean, median, and mode for the maximum number of features that we were able to change and still return a correct prediction in Orange, when considering all initially correctly predicted sources. We also included the count for the mode,

the total number of features in each test set, and the number of source IPs that were originally correctly identified. As can be seen in the Tables, the mean tends to be considerably higher than the mode, even in case where the mode is low and the mode count is high. For example, in Table 4.23, when just the training set is filtered in the *at least 6 days* tests, the KNN values on Tuesday show a mode of 0, a mode count of 30 (out of 76 sources), but a mean of 192. That the mean is so much higher than the mode indicates that there are some outliers in terms of how many destination IPs we were able to change and still have Orange predict the originally correct source. It can also be seen that, when using KNN, each day had a mode of 0, with mode counts that were between 29% and 50% of the tested sources, with a mean of 40.4%. The implies that for about 40% (on average) of the sources changing the number of visits for just one of the destination IPs resulted in a different prediction for the source. For these sources, it would be quite difficult for an adversary to subvert those identities without being detected. :CM EDITED

When filtering for src/dst pairs that were observed on at least six different days during the week, KNN returned a mean accuracy of 54.3%, and 34.5% for the LL analysis using a lifetime of six days. Comparing Tables 4.21 and 4.23, we see that the mean features (that could be changed and still allow sources to be correctly identified) decreased considerably for the KNN classier, and the median values became zero for each day of the week. The mode for both analyses was zero, but with the *at least 6 days* analysis, the mode count was usually quite higher, accounting for over half the sources. For the NB classifier, the comparison was similar, albeit less dramatic. The mean and median values usually decreased, indicating that fewer changes could be absorbed. The mode actually increased, though the mode counts were sufficiently small, so the fact that the modes increased is not very significant.

Perhaps a better comparison would be KNN and NB from the analysis using a six-day LL filter, since the accuracies were similar (NB had 34.8% and KNN yielded 34.5%). As can be seen in Table 4.21, NB has significantly higher mean and median values. Additionally, the mode values for NB, while not high, are also usually greater than zero, and also have a small mode count. Thus, we could say that the profile is stronger when using the NB classifier than when using KNN, as more features could change and still result in a correct identification.

Data	*Test*	*Day*	*mean*	*median*	*mode*	*mode cnt*	*features*	*sources*
Flow Data: LL Filter > 6 days Train Set	NB	Thu	937	349	11	1	11,381	64
		Fri	872	278	5	2	10,478	68
		Sat	128	83	88	3	3,362	56
		Sun	164	79	0	3	2,585	76
		Mon	1,250	527	11	1	11,775	35
		Tue	895	396	77	2	11,900	87
	Tree	Thu	648	240	0	6	11,381	27
		Fri	419	102	0	6	10,478	22
		Sat	215	54	0	4	3,362	32
		Sun	99	49	0	8	2,585	36
		Mon	339	147	0	6	11,775	23
		Tue	372	120	0	10	11,900	34
	KNN	Thu	156	28	0	25	11,381	65
		Fri	259	2	0	28	10,478	56
		Sat	74	6	0	28	3,362	60
		Sun	51	3	0	38	2,585	79
		Mon	287	59	0	25	11,775	76
		Tue	424	79	0	45	11,900	137

Data	*Test*	*Day*	*mean*	*median*	*mode*	*mode cnt*	*features*	*sources*
Flow Data: LL Filter > 6 days Train/Test	NB	Thu	170	51	1	6	2,490	156
		Fri	156	66	0	6	2,295	146
		Sat	58	35	4	6	1,276	81
		Sun	65	32	7	5	1,214	106
		Mon	146	73	5	3	2,375	74
		Tue	191	76	2	8	3,218	202
	Tree	Thu	74	33	0	17	2,490	67
		Fri	84	39	0	15	2,295	59
		Sat	71	28	0	14	1,276	49
		Sun	62	23	0	19	1,214	68
		Mon	199	67	0	8	2,375	45
		Tue	109	30	0	33	3,218	89
	KNN	Thu	53	7	0	56	2,490	134
		Fri	57	11	0	37	2,295	91
		Sat	30	6	0	30	1,276	81
		Sun	32	0	0	60	1,214	114
		Mon	66	8	0	43	2,375	108
		Tue	88	10	0	100	3,218	231

Table 4.21: Flow: statistics for the max number of features that can change and still result in a correct source identification, retaining only LL sources with sliding-window lifetimes > 6 days. Also shown are total number of features and number of sources that were originally identified.

Data	*Test*	*Day*	*mean*	*median*	*mode*	*mode cnt*	*features*	*sources*
Flow Data: $\geq$ 5 diff days Train Set	NB	Thu	1,193	382	50	2	11,381	55
		Fri	1,089	275	13	2	10,478	83
		Sat	167	100	47	3	3,362	78
		Sun	130	69	1	3	2,585	92
		Mon	1214	313	39	2	11,775	47
		Tue	987	320	24	2	11,900	80
	Tree	Thu	623	333	0	5	11,381	42
		Fri	491	184	0	11	10,478	55
		Sat	160	82	0	12	3,362	47
		Sun	145	56	0	17	2,585	59
		Mon	463	241	0	6	11,775	33
		Tue	362	160	0	12	11,900	43
	KNN	Thu	477	87	0	36	11,381	107
		Fri	234	55	0	39	10,478	134
		Sat	96	3	0	51	3,362	113
		Sun	62	2	0	60	2,585	125
		Mon	372	47	0	37	11,775	100
		Tue	483	95	0	29	11,900	107
Data	*Test*	*Day*	*mean*	*median*	*mode*	*mode cnt*	*features*	*sources*
Flow Data: $\geq$ 5 diff days Train/Test	NB	Thu	137	45	1	8	1,873	179
		Fri	129	42	0	10	1,883	177
		Sat	78	38	1	5	1,374	101
		Sun	77	37	10	5	1,380	124
		Mon	130	64	0	5	1,892	99
		Tue	119	39	0	10	1,847	199
	Tree	Thu	64	22	0	31	1,873	89
		Fri	67	33	0	30	1,883	99
		Sat	45	23	0	13	1,374	58
		Sun	76	36	0	24	1,380	96
		Mon	133	61	0	13	1,892	60
		Tue	60	28	0	37	1,847	11
	KNN	Thu	63	5	0	89	1,873	204
		Fri	51	9	0	87	1,883	211
		Sat	27	2	0	58	1,374	130
		Sun	29	0	0	92	1,380	158
		Mon	44	3	0	77	1,892	158
		Tue	40	0	0	107	1,847	213

Table 4.22: Flow: statistics for the max number of features that can change and still result in a correct source identification, retaining only sources active for $\geq$ 5 days. Also shown are total number of features and number of sources that were originally identified.

Data	*Test*	*Day*	*mean*	*median*	*mode*	*mode cnt*	*features*	*sources*
Flow Data: $\geq$ 6 diff days Train Set	NB	Thu	926	458	0	1	11,381	31
		Fri	1,084	535	26	2	10,478	44
		Sat	305	140	2	2	3,362	74
		Sun	177	100	5	2	2,585	74
		Mon	1,016	531	26	2	11,775	46
		Tue	1,135	506	0	1	11,900	43
	Tree	Thu	536	226	0	5	11,381	34
		Fri	413	163	0	10	10,478	32
		Sat	206	106	0	41	3,362	65
		Sun	142	46	0	12	2,585	52
		Mon	935	162	0	6	11,775	30
		Tue	540	285	0	6	11,900	32
	KNN	Thu	313	60	0	29	11,381	82
		Fri	220	24	0	40	10,478	91
		Sat	69	0	0	57	3,362	114
		Sun	52	4	0	51	2,585	115
		Mon	347	150	0	27	11,775	92
		Tue	192	42	0	30	11,900	76

Data	*Test*	*Day*	*mean*	*median*	*mode*	*mode cnt*	*features*	*sources*
Flow Data: $\geq$ 6 diff days Train/Test	NB	Thu	75	42	44	4	1,205	110
		Fri	74	27	4	6	1,201	125
		Sat	76	33	11	6	1,190	115
		Sun	75	48	6	5	1,188	109
		Mon	88	48	0	4	1,210	106
		Tue	86	45	1	5	1,171	119
	Tree	Thu	72	30	0	16	1,205	69
		Fri	65	35	0	19	1,201	84
		Sat	52	24	0	22	1,190	90
		Sun	72	36	0	25	1,188	93
		Mon	62	16	0	33	1,210	89
		Tue	43	26	0	27	1,171	95
	KNN	Thu	16	0	0	77	1,205	143
		Fri	21	0	0	90	1,201	156
		Sat	14	0	0	91	1,190	160
		Sun	24	0	0	92	1,188	149
		Mon	25	0	0	99	1,210	172
		Tue	30	0	0	94	1,171	162

Table 4.23: Flow: statistics for the max number of features that can change and still result in a correct source identification, retaining only sources active for $\geq$ 6 days. Also shown are total number of features and number of sources that were originally identified.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions and Future Work

This thesis sought to investigate user network behavior profiles that associate a particular source IP address with the number of visits to unique destination IPs, and whether such profiles can be applied to network traffic from an academic building at NPS to identify network users. Further, how would using datasets containing DNS queries compare with datasets containing destination IPs, in terms of identification accuracy.

Others, such as Banse *et al.* [5] and Yang [6] had achieved success in using DNS and destination IPs for user identification, but both involved home users or residential housing at a university. We wanted to investigate how these techniques would fare on data that more closely approximates a government or corporate office. User network behavior differs based on the user's location [17], and it makes sense that traffic from an office building might contain fewer unique markers, as users may regularly perform network activity while at home, but may do so less often while at work. While traffic from the Computer Science Department at NPS is not a perfect replacement for traffic from a government or corporate office, we believe it is a reasonable substitute, at least for initial analyses.

We began by collecting pcaps of network traffic from the Computer Science Department at NPS, separating the pcaps into DNS traffic and IP flow traffic. We parsed the pcaps using C++ with libpcap, extracting the source IPs and destination IPs from the flow data, and used C++ maps to keep track of the number of times each source IP issued a TCP SYN for each destination IP, within the course of a 24-hour period. After performing tf-idf on the numbers of SYNs issued by each source, we formatted and wrote the information to a "tab" file. The tab files were used as input for the data-mining software package Orange [20], which computed accuracies using NB, Classification Tree, and KNN. The procedure was repeated for the DNS datasets, except that instead of recording number of SYNs, we recorded the number of times each source IP issued a DNS request containing a particular QNAME. We also analyzed the performance when cropping the DNS hostnames, to see if focusing on just the parent domain would affect the identification accuracies.

5.1 Conclusions

The results of our analysis indicate that identification of users based on network traffic is applicable to NPS data, although the accuracy results are lower than when considering residential networks. The results further imply that using DNS queries can yield higher accuracy results than using destination IPs, as was the case for every test we performed. For both the NB and KNN classifiers, the DNS queries returned between 20-25% higher accuracy than using destination IPs, when training on one day and testing on the next.

We also calculated the longevity of an IP flow or DNS query, based on the *lifetime* of the access, i.e., when the access was first seen and when it was last seen, over the course of a week. We then created CDFs of these lifetimes. Analysis of the CDFs indicates that roughly half of all TCP SYNs from a given source IP were only issued during a single network session, and were never issued again during the following seven-day period by that same source. Over half of the DNS queries issued by a given source were never queried again by that source in the next seven days. Roughly 40% of all TCP SYNs in the dataset were never sent again by *any* source during the seven days following the initial web visit 60% of all DNS queries were only issued during a single network session, and were never issued again by *any* source over the next seven days. About 10% of SYNs were regularly issued by the same source over the course of seven days. Similarly, about 10% of DNS queries were regularly issued by the same source over the seven-day period. Approximately 30% of SYNs were regularly issued by at least one source for at least seven days during the assessment period. About 10% of the DNS queries issued by any source were regularly issued again by at least one source during the next seven days.

Lifetime Filters. On unfiltered datasets, training on one day and testing on the immediately following day, the NB classifier returned mean accuracies of 27.1% for IP flow data, 46.5% for DNS data, and 43.4% for the modified DNS data. The KNN classifier yielded means of 13.5% for flow data, 30.9% for DNS, and 40.0% for the modified DNS data. For unfiltered datasets, using DNS returned significantly higher identification accuracies than IP flow data. The DNS hostname cropping technique resulted in slightly lower accuracies for NB, but significantly higher results for KNN, though the KNN scores were still lower than NB for these tests.

Short-lived Filters. When the very short-lived accesses were removed from the datasets, the identification accuracies for flow and DNS datasets improved. The largest gains occurred when both the training and test sets were filtered to omit short-lived source/destination IP pairs or source IP/DNS hostname query pairs. NB returned accuracies of 33.8% for flow data, 52.0%

for DNS, and 47.6% for modified DNS. The KNN classifier achieved 18.6% for flow, 39.2% for DNS, and 44.8% for the modified DNS. As with the unfiltered datasets, we saw the NB was the more accurate than KNN, and that KNN had a larger increase in accuracy when using the modified DNS. When omitting short-lived src/dst pairs or src/hostname pairs from just the training set, the increases were still present, but were more slight for NB, though KNN still retained an 8.3% increase for DNS. Tests that omitted short-lived destination IPs or DNS queries, irrespective of source IP, did not have any significant effect on accuracy for any of the classifiers we used.

Long-lived Filters. We also created datasets in which only the long-lived accesses were retained. Our definition of "long-lived" was those accesses having a lifetime of at least six days. We then reran the trials using five days as the threshold for being considered long-lived. The accuracies for all classifiers increased dramatically when filtering both training set and test set for src/dst pairs or src/hostname pairs using threshold of six days. For these tests, NB returned mean accuracies of 34.8% for flow data, 59.4% for DNS data, and 54.6% for modified DNS. The KNN classifier performed quite well, yielding 34.5% for flow data, 59.5% for DNS data, and 63.1% for modified DNS. Interestingly, when only the training sets were filtered, the accuracies often decreased significantly as compared to the short-lived analysis. This seems to imply that there are SYNs and DNS queries that are useful for identification on a day-to-day scale, but that may not persist over the course of a week. The cost of the increased accuracy was that fewer source IPs were present in the filtered datasets, which means that many source IPs were not able to be classified.

At Least X Days. Another version of filters we explored were those that looked at the number of different days during the week a particular destination IP or DNS query was observed. When filtering both training and test sets for src/dst or src/host pairs that were active for X days, the largest gains for flow data occurred when $X = 5$ for NB (42.7%), and $X = 6$ (54.3%) for KNN. For DNS data, the maximum accuracies were with $X = 6$ for both NB (69.6%) and KNN (86.5%). With datasets containing the modified DNS hostnames, the results were similar to the unmodified DNS. As with the filters using lifetimes, the cost of this increased accuracy was that the number of source IPs in the datasets decreased as the filters became more stringent. The implication is that are are some sources that engage in somewhat regular network behavior, which can aid in the identification of those particular sources.

Multiple Daily Filters. The final version of tests involved using multiple daily instances (MDI) in the training set, but still testing on the immediately following day. For unfiltered MDI, the highest accuracies were from the Classification Tree with training sets consisting of four days. The Tree classifier returned accuracies of 40% for IP flow data, 76.5% for DNS, and 75.9% for modified DNS. When filtering the datasets to only retain long-lived access, we again saw that the largest gains were when both training and testing sets were filtered for src/dst pairs or source/hostname pairs. KNN yielded the highest accuracies, with six days in the training set, achieving 71.9% accuracy for flow data, 90.1% for DNS, and 87.9% for modified DNS.

AS Statistics. We also looked at the ASNs for the flow data, comparing the popularity of the ASs for all destination IPs, just the very short-lived destination IPs, and just the very long-lived destination IPs. Our findings indicated that the large ASs, such as Amazon, Google, Microsoft, and Akamai, were some of the most common ASs, regardless of whether we considered short-lived, long-lived, or all accesses.

Profile Strength. Lastly, we explored the strength or weakness of the profiles, for selected analyses, by looking at the maximum number of features that could be changed and still correctly identify the source. While more testing should be done on this, our initial findings seem to suggest that for our profiles, NB was able to produce more resilient profiles than KNN.

5.2 Future Work

This thesis provided some groundwork for manipulating user network behavior profiles, and demonstrated that certain techniques can be applied to non-residential data, such as traffic from an academic department at NPS. There is clearly much more work to be done in this field.

User to Source IP Correlation. It is likely that ephemeral source IPs played a role in lowering our accuracy. Some of these source IPs certainly would have changed over the course of the week we were observing. Additionally, the the lists of short-lived and long-lived src/dst and src/hostname pairs to either omit or retain were created using a one-week sliding window, which means that the whole period covered by the sliding windows was a two-week period. It is possible that a significant number of source IPs changed during that period. If a user visited a destination regularly during the two week period, but the user's source IP changed, then that src/dst pair might register a lower lifetime than it would have. It would be interesting to correlate (and anonymize) users to the source IPs, accounting for changes in the IPs, to see what effect this has on accuracy.

The short-lived lifetimes, however, were likely not affected too much by ephemeral IPs. The SL lifetimes were considerably small, so the ephemeral IPs would have had to change very frequently to significantly affect the results. It is possible that a user could have accessed a destination during one session, and then returned to that destination later, but with a different source IP. In that case, we would have incorrectly identified and omitted one SL src/dst pair, and retained a different src/dst pair (but the same user), but with a shorter lifetime. While we do not believe such a case would have a significant effect on identification accuracy, the only way to know would be with source IP to user correlation.

Refine Threshold for Short-lived and Long-lived Lifetimes. There are also variations that could be done on our definition of long-lived and short-lived. Our short-lived lifetimes were all less than 25 minutes, as we wanted to focus on accesses that were performed only during a short network session, and then never repeated. It might be useful to increase the time for an access to be short-lived. For example, accesses with lifetimes of less than a day might not be included in both a training set and a test set, and so are just noise. Additionally, our long-lived lifetimes of six days and, subsequently, five days, might be too stringent. If a source makes the same network access for four days in a row, takes a three day break, and then makes the same access again, a five-day threshold would miss this activity. It might be interesting to combine the idea of long-lived accesses with the number of times in a week a destination was accessed. For our analysis, these methods were separate.

Conduct Further Testing with MDI. More complete tests with the multiple daily instances could be performed. Due to limited time, we were only able to create and analyze selected data. A two-day training set was always Wednesday and Thursday, a three-day training set was always Wednesday, Thursday, and Friday, and so on. It would be interesting to see the results of training on every two adjacent days.

Enhanced DNS Hostname Cropping. There are also enhancements that could be made to the DNS modifications, in which we cropped the hostname from the QNAME field. In particular, it might be possible to focus on hostnames that were likely issued by a user. Perhaps a more realistic goal would be to omit many of the DNS queries that were automatically sent from an advertisements, embedded URLs, or DNS Prefetching. One method might be to identify and remove queries containing CDN domains, such `edgekey.net` or `edgesuite.net`, that indicate the Akamai CDN. It is unlikely that a user would issue a query for such a hostname. The potential pitfall with this technique would be that if a user regularly visits a website that generates

such DNS queries, those queries might themselves help identify the source. Additionally, some DNS queries that are sent as a result of DNS Prefetching have forms that are not unlike normal user DNS queries.

With both the modified and unmodified DNS queries, the types of queries we focused on were A, AAAA, TXT, SRV, and MX records, thinking that these types might reveal uniqueness in user DNS requests. It might be useful to look at just A records, or other variations to see if there is any effect on identification accuracy.

Omit Uncommon Sources and Destinations. Finally, perhaps it would be of value to omit destination IPs and DNS queries from the test set, if those IPs and queries are not present in both the training set. A destination IP or DNS query that is not on both the training and test set is not likely to contribute to the classification.

Conduct Further Tests of Profile Strength. The best performing profiles in our analyses were those created with DNS queries. Due to time constraints, we were not able to test how strong those DNS profiles were. It would be interesting to see if those more accurate DNS profiles were also more resilient.

REFERENCES

[1] E. Orimo, H. Koike, T. Masui, and A. Takeuchi, "Analysis and Evaluation of Recommendation Systems," in *Proc. of the 2007 conf. on Human interface: Part I*, Beijing, China, 2007, pp. 144–152.

[2] J. Geary. (2012, April 23) DoubleClick (Google): What is it and what does it do? *Guardian*. [Online]. Available: http://www.theguardian.com/technology/2012/apr/23/doubleclick-tracking-trackers-cookies-web-monitoring

[3] G. Greenwald, E. MacAskill, and L. Poitras. (2013, June 9) Edward Snowden: The whistleblower behind the NSA surveillance revelations. *Guardian*. [Online]. Available: http://www.guardian.co.uk/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance

[4] C. Savage. (2013, Feb. 28) Soldier Admits Providing Files to WikiLeaks. *New York Times*. [Online]. Available: http://www.nytimes.com/2013/03/01/us/bradley-manning-admits-giving-trove-of-military-data-to-wikileaks.html

[5] C. Banse, D. Herrmann, and H. Federrath, "Tracking users on the internet with behavioral patterns: Evaluation of its practical feasibility." in *SEC*, ser. IFIP Advances in Inform. and Commun. Technology, D. Gritzalis, S. Furnell, and M. Theoharidou, Eds., vol. 376. Heraklion, Greece: Springer, 2012, pp. 235–248.

[6] Y. C. Yang, "Web user behavioral profiling for user identification," *Decis. Support Syst.*, vol. 49, no. 3, pp. 261–271, June 2010.

[7] F. L. Greitzer, L. J. Kangas, C. F. Noonan, A. C. Dalton, and R. E. Hohimer, "Identifying At-Risk Employees: Modeling Psychosocial Precursors of Potential Insider Threats," in *Proc. of the 2012 45th Hawaii Int. Conf. on System Sciences*, ser. HICSS '12. IEEE Comput. Society, 2012, pp. 2392–2401.

[8] CSO Magazine, USSS, CERT, and Deloitte, "2011 CyberSecurity Watch Survey," CERT Program, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, 2011.

[9] P. Reidy and K. Randal, "Combating the Insider Threat: Real World Lessons," presented at the RSA Conf. 2013, San Francisco, CA, 2013.

[10] G. Silowash, D. Cappelli, A. Moore, R. Trzeciak, T. J. Shimeall, and L. Flynn, "Common Sense Guide to Mitigating Insider Threats," CERT Program, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU/SEI-2012-TR-012, Dec. 2012.

[11] A. P. Moore, M. Hanley, and D. Mundie, "A Pattern for Increased Monitoring for Intellectual Property Theft by Departing Insiders," CERT Program, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU/SEI-2012-TR-008, April 2012.

[12] W. R. Claycomb, C. L. Huth, L. Flynn, D. M. McIntire, and T. B. Lewellen, "Chronological Examination of Insider Threat Sabotage: Preliminary Observations," *J. of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applicat. (JoWUA)*, vol. 3, no. 4, pp. 4–20, Dec. 2012.

[13] A. Cummings and R. Trzeciak, "Insider Threats and Security Trends: Lessons Learned from Actual Insider Attacks," CERT Program, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, 2010.

[14] J. Klensin, "Role of the Domain Name System (DNS)," RFC 3467 (Informational), Internet Eng. Task Force, Feb. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3467.txt

[15] P. V. Mockapetris, "Domain Names - Implementation and Specification," RFC 1035 (INTERNET STANDARD), Internet Eng. Task Force, Nov. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[16] D. Herrmann, C. Gerber, C. Banse, and H. Federrath, "Analyzing characteristic host access patterns for re-identification of web user sessions," in *Proc. of the 15th Nordic Conf. on Inform. Security Technology for Applicat.*, Espoo, Finland, 2012, pp. 136–154.

[17] F. Giroire, J. Chandrashekar, G. Iannaccone, K. Papagiannaki, E. M. Schooler, and N. Taft, "The cubicle vs. the coffee shop: behavioral modes in enterprise end-users," in *Proc. of the 9th Int. Conf. on Passive and Active Network Measurement*, Cleveland, OH, 2008, pp. 202–211.

[18] S. McKinney and D. S. Reeves, "User identification via process profiling: extended abstract," in *Proc. of the 5th Annu. Workshop on Cyber Security and Inform. Intell. Research: Cyber Security and Inform. Intell. Challenges and Strategies*, Oak Ridge, TN, 2009, pp. 51:1–51:4.

[19] A. Udoeyop, "Cyber Profiling for Insider Threat Detection," M.S. thesis, Dept. Comput. Sci., Univ. of Tennessee, Knoxville, Aug. 2010.

[20] J. Demšar, B. Zupan, G. Leban, and T. Curk, "Orange: From Experimental Machine Learning to Interactive Data Mining," in *Proc. of the 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases*. Faculty of Comput. and Inform. Science, Univ. of Ljubljana, 2004, pp. 537–539.

[21] A. Gulbrandsen and P. Vixie, "A DNS RR for specifying the location of services (DNS SRV)," RFC 2052 (Experimental), Internet Eng. Task Force, Oct. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc2052.txt

[22] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi, "DNS Extensions to Support IP Version 6," RFC 3596 (Draft Standard), Internet Eng. Task Force, Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3596.txt

[23] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY: Cambridge Univ. Press, 2008.

[24] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann Publishers Inc., 2005.

[25] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ: Prentice Hall Press, 2009.

[26] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," in *Proc. of the 2007 Conf. on Emerging Artificial Intell. Applications in Comput. Eng.: Real Word AI Syst. with Applicat. in eHealth, HCI, Inform. Retrieval and Pervasive Technologies*, 2007, pp. 3–24.

[27] J. Demšar, B. Zupan, G. Leban, and T. Curk, "Orange Documentation v2.7.1," http://orange.biolab.si/docs/latest.

[28] C. O'Neil. (2013, April 4) K-Nearest Neighbors: Dangerously Simple. [Online]. Available: http://mathbabe.org/2013/04/04/k-nearest-neighbors-dangerously-simple/

[29] E. Cohen and H. Kaplan, "Prefetching the means for document transfer: A new approach for reducing Web latency," *Comput. Networks*, vol. 39, no. 4, pp. 437 – 455, 2002.

THIS PAGE INTENTIONALLY LEFT BLANK

Referenced Authors

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California