CHARACTERIZING UNIVERSITY NETWORK USAGE WITH ACTIVE DIRECTORY EVENT LOGS

BY

ALEX MITSDARFER

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Associate Professor Michael Bailey

ABSTRACT

In this thesis, we investigate a university network that uses Active Directory as its authentication system. We get an understanding of the network by analyzing Windows event logs generated at Active Directory domain controllers. We want to see what network activity looks like as a first step in identifying and modeling network lateral movement. We characterize network activity, access behavior, most frequent events encountered, and domain controller usage. We find that the data, covering a week's time, supports multiple trends. The number of events encountered increases from morning to noon and decreases after mid-afternoon. Weekend activity is lower than during weekdays. Over the week of user-generated events, about 85% create 1,000 events or less. Less than 5% of users create more than 10,000 events. The top five events encountered are associated with user sessions (i.e., login, logout, authentication) or Kerberos ticket requests. Most events are generated at the Urbana Domain Controllers. The second largest number of events (although about 15 times smaller) are generated at the DCs that serve only WiFi and VPN.

To my parents, my siblings, and my wife, for their love and support.

ACKNOWLEDGMENTS

I would like to thank Professor Michael Bailey for his guidance and counsel. I want to also thank Technology Services at UIUC for their generous help and cooperation. Special thanks to Erik Coleman, Jon Gillen, and Charles Geigner for their dedication and effort.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

For many sophisticated cyber attacks, breaching a network is only the first step. Post-exploitation, an attacker may want to expand control to other network resources [1, 2, 3]. In order to locate a target system and accomplish its goal, an attacker relies on moving around in a network, undetected, until the target is found. An adversary cannot always carry out a sophisticated attack by directly compromising the target system. An attacker must instead compromise one system on the network that is vulnerable, learn about the breached network, look for vulnerabilites on other systems, and find and compromise the target [4, 5, 6]. Trend Micro [5] organizes these network attacks into 6 stages: Intelligence Gathering, Point of Entry, Command and Control Communication, Lateral Movement, Maintenance, and Data Exfiltration. There is some overlap between the stages, and some stages might involve repeating previous stages. In this thesis, we look specifically at the threat of lateral movement. According to [5], the three goals of lateral movement are:

1. obtain escalated privileges within the target network,
2. learn about the target network through observation, and
3. gain access to other machines within the network.

We focus our attention on the latter two goals.

For a defender, it is crucial to minimize and detect these types of attacks. It can be difficult, however, to know if a stealthy adversary is moving around in the network. One way for a defender to gain an advantage is by thoroughly

knowing the network he is protecting. This becomes problematic when the network has thousands or hundreds of thousands of nodes.

One approach is to actively monitor the network. The defender can build a base-line for what would be considered normal behavior, and continue to monitor for anomalies. If network activity deviates from expected, the defender can investigate the cause of the alarm.

In order to study lateral movement, we want to first understand normal user movement. In this thesis, we investigate a university network that uses Active Directory as its authentication system. We get an understanding of the network by analyzing user logins, logouts, usage patterns, and how Kerberos tickets are distributed. We investigate event logs produced at Active Directory controllers to identify behavior. The behaviors we are interested in include access patterns and usage distributions.

1.1.1 Active Directory

Active Directory (AD) is a Microsoft service for managing Windows domain networks. Active Directory centralizes user and resource management. Network administrators can add and modify information about users and groups, computers and printers, and applications and services efficiently from the central repository. This information can then be distributed and made available for the network [7]. Network admins can allow a user to access resources without individually configuring each resource. For example, a user is allowed to log into any computer in a computer lab, and the admin does not have to create a local account for the user at each computer. An Active Directory Domain Services (AD DS) server is also called a domain controller. It is the entity that authenticates and authorizes the users and computers on the network, as well as enforcing their access policies. AD supports multiple protocols: DNS, Lightweight Directory Access Protocol (LDAP), and Kerberos. LDAP is used primarily for internal AD processes such as clients downloading schemas and retrieving policies. Kerberos is a protocol for secure user authentication, even on an insecure network [8, 9].

1.1.2 Active Directory Authentication

Windows NT LAN Manager (NTLM) and Kerberos protocols can be used to authenticate a user in Active Directory. Kerberos is the preferred method, but cannot always be used. Kerberos cannot be used if a domain is running Windows NT 4.0 or older, if the client is using an IP address to connect to an AD service rather than a host name, if the client is accessing a resource that is not a member of the AD domain, or if the resource does not support the Kerberos protocol [10].

1.1.3 NTLM

NTLM is a challenge-response protocol for authenticating a user and a computer. The NTLM protocol involves the use of LAN Manager (LM) hashes and Windows NT (NT) hashes. Details about these hashes are explored in [10, 11]. NTLM can be used to authenticate a local user account on a computer or a domain account on a domain (such as through Active Directory) [12]. As of the release of Windows 7 and Windows Server 2008 R2, session security policy is set to require a 128-bit minimum encryption for clients and servers [13]. Older versions of NTLM use 40-bit and 56-bit keys.

There are two versions of the protocol: interactive and noninteractive. Interactive is used when the user wants to authenticate with a computer. Noninteractive is when the user is already logged into the computer and wants to access a resource. The following describes the "three-way handshake" for the NTLM authentication protocol [14, 10, 15].

1. This step is exclusive to interactive authentication. The client wants to access a computer. The client provides a domain name, username, and password to a client computer. The password is hashed and the original password is discarded.

2. The client requests a challenge from the authentication server, providing his username in plaintext.

3. The server creates a 16-byte random number to use as the challenge. This challenge is sent to the client.

4. The client receives the challenge and computes a response. The random number is encrypted with the hashed password. The client sends the response to the server.

5. The server receives the client's challenge response. It then forwards this response, the original challenge, and the username to the domain controller (DC).

6. The domain controller has a database of usernames and password hashes. The DC looks up the user's stored hash and encrypts the challenge with this hash. It then compares its result to the response that the user computed. If they are the same, the user is successfully authenticated.

1.1.4 Kerberos

The Kerberos protocol for authentication involves three parties: a client, a server (resource), and an authentication server called the Key Distribution Center (KDC).

Kerberos relies on long-term and short-term cryptographic keys for encryption and decryption. Long-term keys are used for verifying user, system, and service identities. These keys are derived from passwords. When a user account is created in Active Directory, a key, derived from their password, is stored in a KDC database. When the user wants to log in, they supply their password and the user key is created [9]. Short-term keys are used for communication in which the session is not expected to last as long. As we will see later, session keys that are used for temporary client-service communication are short-term keys.

The KDC's role is also to avoid each user needing to maintain keys for each server, and for each server to maintain keys for each user. What follows is the protocol for how a user obtains access to a network resource using Kerberos version 5 [9]. This is also illustrated in Figure 1.1. Note that the TGS is illustrated as running on the KDC, but this is not a requirement.

1. When the user wants to access a service, such as a network printer, the user requests permission from the KDC to access the Ticket-Granting-Server (TGS). When Kerberos Preauthentication is enabled, the user sends his username and a timestamp encrypted with his own encryption

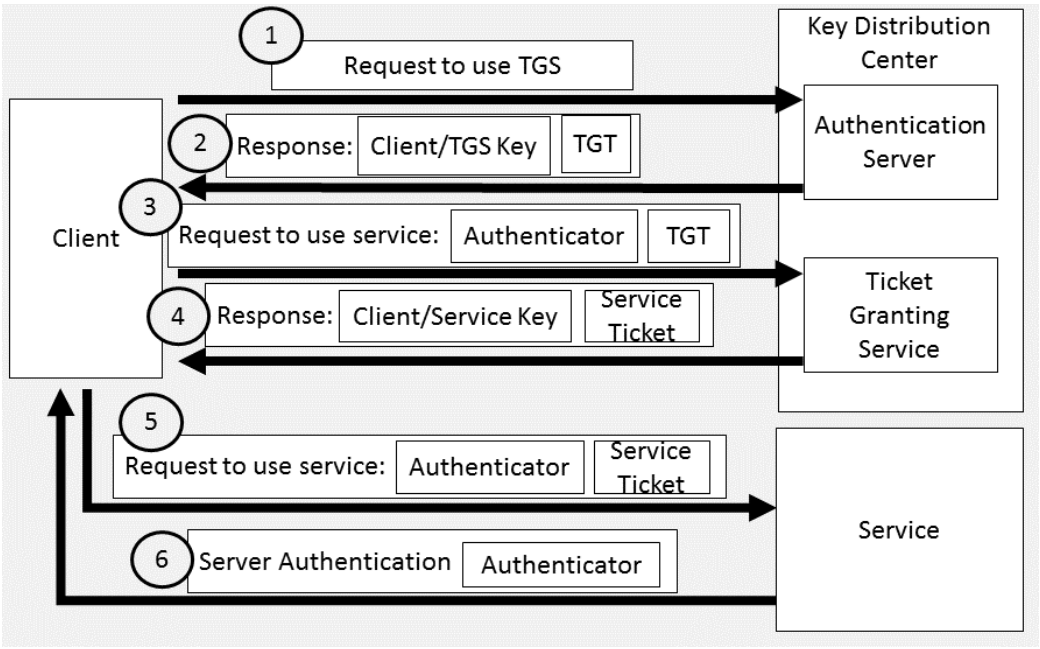

Figure 1.1: Kerberos Protocol

key, the long-term key, as evidence of his identity [8]. The Kerberos server looks up the client in its database (only checks if the client exists). The KDC also has a copy of the user's key in its database, which it uses to decrypt the timestamp to verify the user's identity.

If Preauthentication fails or is disabled, an alternative, similar step takes place instead. The user first sends a request to the KDC that initiates the authentication process. To confirm the user's identity, the KDC responds to the user with a message encrypted with the user's private key, which the KDC has on record. Only the user's key can decrypt this message. By decrypting this message and continuing with the protocol, the KDC confirms the user owns his key, confirming his identity. This completes the authentication step. The KDC also identifies if the user is authorized to use the requested resource through permission policies.

2. A session key (SK1) is generated for use between the client and the TGS. The Kerberos server responds to the client with two messages. One message contains information about the TGS, a timestamp, a ticket lifetime, and SK1, and is encrypted with the clients private key. The second message is the TGT and contains the clients information,

timestamp, network information, TGT lifetime, and SK1. The TGT is encrypted with the TGS private key, which the client does not know. The user must provide this TGT whenever he wants to request, from the TGS, access to other network resources.

3. The client decrypts the message and recovers SK1. The client now builds two messages to send to the TGS. It first builds a data structure called the Authenticator, containing the clients information and timestamp. The first message contains unencrypted request information (the desired resource) and requested lifetime of the ticket. The second message contains the Authenticator, encrypted with KS1, and TGT, still encrypted with the TGS private key (from the Kerberos authentication server). The TGT is used to request service tickets (for services such as Microsoft Exchange, network drives, or network printers [16]) from the TGS. These two messages are sent to the TGS.

4. The TGS does a KDC database lookup to make sure the requested service exists. The TGS uses its own private key to decrypt the TGT. The TGT contains SK1, so the TGS now uses SK1 to decrypt the Authenticator. Information from the Authenticator is validated with the TGT. The TGS then generates a session key (SK2) for the client and the resource to use. It sends two messages to the client: the first contains SK2 and client information, and is encrypted with SK1. The second is a resource service ticket that contains the clients information, network information, timestamp, lifetime, and SK2, which is encrypted with the resources private key. A service ticket is good only for the particular service that was requested by the user. Whenever the user wants to access the service for which the ticket is specified, the user must provide this service ticket.

5. The client decrypts the client message with SK1 to obtain SK2. The client is now ready to talk to the resource. It sends two messages: the first is another Authenticator that contains the clients information and timestamp, encrypted with SK2. The second message is the resource service ticket obtained from the TGS (still encrypted with the resource private key).

6. The resource essentially repeats the steps that the TGS performed. It

decrypts the resource service ticket with its own private key to obtain SK2. It uses SK2 to decrypt the Authenticator and validate the user's information. The service can use the contents of the decrypted service ticket to confirm the user named in the ticket matches the user trying to use the ticket. The client is now authenticated to use the resource.

7. The following steps are optional for Kerberos. The resource then sends its own Authenticator message to the client to confirm its identity, encrypted with SK2.

8. The client receives the resource's Authenticator message, decrypts with SK2, and can confirm that the resource is the intended resource. The client now confirmed the identity of the service.

To summarize, the user password hash/key is used to obtain a TGT, a TGT is used to obtain service tickets, and service tickets are used to gain access to services.

The client caches the TGT and any resource tickets. TGTs and service tickets have a default lifetime of 10 hours [8, 17, 18]. The client can then check its cache for resource credentials (and if not found or expired, the TGT) before going through the whole Kerberos protocol. From the perspective of the client, the TGT is essentially just another ticket that allows access to a resource. From the KDCs perspective, the TGT is a way to reduce the number of ticket requests, and therefore reduce network communication and processing.

1.2 Lateral Movement

Imagine a situation in which an attacker has gained access to one computer on a network through some vulnerability (e.g., code injection attack) or user error (e.g., malicious e-mail attachment). If the attacker wants to expand his access, he may wish to gain access to other computers or resources in the network. This is also known as network lateral movement. For example, the attacker may be able to compromise a local account on a regular workstation. That account might not have gone through the Kerberos protocol and gained access to certain resources (i.e., possessing service tickets). Alternatively, the

attacker may want to log into a particular resource using a particular account (e.g., access a network folder as the user Dan Smith). There are multiple attacks that can be performed to move laterally in a network.

What is the motivation for lateral movement? According to the works of many security companies [4, 6, 19], lateral movement is useful to an adversary for many reasons. They can gather information about a network or systems on the network, gain access to specific files or credentials, and even execute code on target systems. Lateral movement is an important step for an adversary that has a specific long-term goal or a target that is hard-to-reach from an outside network. Such an adversary could be an advanced persistent threat (APT). An APT with sufficient resources and motivation would be willing to infiltrate a network and move laterally until the desired target system is found.

1.3 Attacks on Active Directory

1.3.1 Identity Snowball Attacks

Identity snowball attacks are a category of attacks that describe network lateral movement. While not specific to Kerberos, an identity snowball attack, as detailed in [20], is described as follows. An attacker leverages a user's credentials to gain access to another resource, and the obtained resource allows access to another resource, and those resources allow access to another resource, and so on. The first user's credentials are obtained at a compromised machine. The credentials obtained are at an elevated level such that access to other resources is possible. For example, user Alice is a network administrator. Alice's machine is compromised and her credentials are obtained by an adversary. The adversary can now access Bob's machine using Alice's credentials. Now Bob's credentials are compromised by the adversary and can be used to log into Carol's machine. This repeats, and the adversary is therefore moving laterally in the network.

1.3.2 Pass the Hash Attack

User passwords are stored as a hash or a key, as described previously, and kept in memory. This is to avoid the need to continually prompt the user for a password on each Active Directory transmission, also called Single Sign-On (SSO). The user can log in once using his password, and this information will be kept in memory for a while without the user needing to reenter the password. The hashes in memory serve the same purpose as a password. From an attacker's perspective, obtaining a hash is nearly as good as obtaining a plain text password. This is because a hash can be used to authenticate a user, just like a password.

In a "pass-the-hash" attack, the attacker obtains a user's password hash and impersonates the user. If a user's machine is compromised, the attacker can read the user hash from memory. The hash can be obtained from the Windows Local Security Authority (LSA) service. The LSA service handles password hashes (such as NTLM hashes stored in the Security Accounts Manager (SAM) and Kerberos hashes and tickets stored in a directory services database). With administrative privilege, memory can be dumped from these regions [21].

The attacker can store the retrieved hash in his own LSA, pretending to be the user. The attacker can now follow the NTLM or Kerberos protocols like normal. A TGT can be requested from the KDC, and service tickets can be retrieved from the obtained TGT.

"Pass-the-hash" refers to using a recovered LM or NTLM hash, and "overpass-the-hash," also called "pass-the-key," refers to using a recovered AES or RC4 key, but the concept of the attack is the same in either case. The user's credentials are stolen and used to follow the Kerberos protocol to obtain a TGT and possibly service tickets.

1.3.3 Pass the Ticket Attack

A ticket, whether crafted or obtained from memory, can be injected into the current session. This means that the ticket is submitted to the TGS or to the service in order to obtain access to the desired service. This behavior is permitted and a Windows API call is available to perform this ticket injection [22].

The attacker can craft their own Kerberos tickets if the AD controller or service is compromised. The attacker can use the stolen hash to create a TGT. To craft a TGT, the hash would need to be obtained from the Kerberos service account (krbtgt), which can be obtained from the LSA of the domain controller. Alternatively (or in addition), the attacker could craft a service ticket. To do this, the hash would need to be obtained from the service account. Compromising the AD controller would be a best-case scenario for the attacker, as private keys of user accounts and services would be accessible. In this study, we focus our attention on what an attacker looks at in a network prior to compromising an AD controller. We look at leaf nodes of a network, which are workstations and services.

User Kerberos tickets (TGT and service tickets) are stored in memory. This is to avoid continually going through the Kerberos protocol for every request to use a network service. With administrative privilege, these tickets can be read from memory. The attacker can inject an obtained ticket into the current session. This means taking the recovered TGT or service ticket and inserting it into the LSA (i.e., on a different computer). The attacker needs to know the username associated with the injected ticket, as well. This injection is done through a Windows API and does not require admin rights. If the TGT is injected, the attacker can then request service tickets using the TGT.

1.4 Research Questions

A future goal is to build models of how an attacker would move laterally in the network. Before we can understand how an attacker can behave on the network, we want to first understand how normal users behave. We want to see what normal looks like on the network.

Active Directory controllers, TGSs, and Kerberos services log events and network transactions. These logs are sent to an aggregation point where they can be collected, stored, and later analyzed. We will analyze the logs from Active Directory controllers that service network requests throughout the entire university campus. Objectives of this research include:

1. Identify the number of unique users, recurrence of users, frequency of users (frequency of logins)

2. Characterize usage over time and where requests are served

3. Discover the number of TGTs and service tickets available

This research is significant because it provides security researchers a better understanding of network usage on a university campus. University IT at this campus can directly use this information to identify and investigate anomolies or unexpected behavior we might find. Security researchers can use the information provided to better understand large networks, including usage patterns and frequency of events.

One of the goals of this work is to use the log data to better understand the network. As a network defender, visualizing the log data is one way to better understand network behavior and see patterns. A manager at the Microsoft Threat Intelligence Center claims that defenders are at a security disadvantage when they think of the network as a list rather than a graph [23]. Having a list of systems to secure is useful, but it is also useful to know how the systems are actively being used. It is easier to detect obvious outliers if normal is well known. More detailed usage patterns are useful for detecting more subtle lateral movement. We hope to provide these insights throughout this thesis.

The described threats directly impact organizations, corporations, universities, and other entities that use internal networks running directory services.

1.5 Contributions

We analyzed Windows event logs produced at a university campus consisting of over 44,000 students and an additional 5,000 faculty and staff [24]. This is one of the largest studies we have seen in terms of user population [25, 26, 27]. The logs were over a week of network activity.

We characterized the activity we saw from users, services, and shares. We described usage over a week, detailed usage based on event codes encountered, looked at daily average usage, and discussed distribution across the domain controllers. We repeat the previous analysis with the filtered data to gain additional insight on network behavior without services and shares. We provide additional insight to the university IT, Technology Services, about

the network usage, such as points of unusually high traffic and authentication failures.

CHAPTER 2

LITERATURE REVIEW

2.1 Machine Learning Detection

The author of [28] uses the network analyzer BRO to detect lateral movement in a network. This work focuses on the Server Message Block (SMB) protocol, although one segment of the work focuses on detecting pass the hash attacks. He uses machine learning to identify what normal and abnormal behavior looks like, so that anomalies can be detected which could indicate an attacker attempting to move laterally in a network. This work does not cover the Kerberos protocol and therefore is lacking detection for pass the ticket attacks. Further, the approach implements policies to protect against these behaviors. This black-list approach is limited by how many attacks and behaviors the defenders can think of. Results from a corporate network dataset indicate that this identification is plausible with relatively low false positives (one per hour).

DExtor [29] is a data mining network analyzer that focuses solely on detecting code. It runs under the assumption that only data is transferred on a network and code is malicious. They use machine learning to differentiate data and code, and place the detector on a live network. DExtor works at the application layer, which is also where Kerberos resides. Their tests indicate high accuracy for detecting code in network traffic and low false positives. It is uncertain if their approach can be performed in real-time. Unfortunately, an attacker wishing to move laterally in a network using Kerberos will continue to correctly follow the Kerberos protocol. The packets transferred will only contain data.

2.2 Analysis Tools

APT-Hunter [30], on the other hand, helps security analysts detect legitimate logins that are carried out by an adversary. APT-Hunter is a visualization tool that analysts can use to identify lateral movement in the form of legitimate-looking logins. It helps visualize links and login patterns that are suspicious, such as desktop-to-desktop connections. In their study, two analysts used APT-Hunter to identify 349 out of 749 total malicious logins (done from a red team) with a false positive rate of 0.005%. This analysis was done offline, so the practicality of APT-Hunter in real-time is uncertain. Further, while the study demonstrates some success, about 53% of malicious logins were missed. With a large enterprise network, manual evaluation is time and resource intensive.

The authors of [31] use reachability graphs to quantify the risk for threats on a network. They calculate a metric as the likelihood that a graph node is reachable from another graph node. Pass the hash is one example of threats they say can potentially be predicted. They evaluate only the performance of this system, so the practicality and accuracy of detection is unknown. This strategy may be too simple by only identifying what nodes are at higher risk based on how many other nodes it connects to. This assumption might not hold true in enterprise networks when an adversary is more likely to target client workstations rather than high-traffic servers.

The authors of [20] created a tool to help network administrators defend their networks. Heat-ray combines machine learning, combinatorial optimization, and attack graphs to help IT make decisions on how to manage their network. They focus on minimizing identity snowball attacks. Heat-ray suggests configurations that eliminate unnecessary network links, reviews the number of users with escalated privileges, removes out-of-date privileges, removes group privilege assignments that are no longer needed or over-encompassing, prevents high-privilege accounts from unnecessary logins, and secures automated script execution. It attempts to do all this while not preventing users from accomplishing their tasks. Their evaluation demonstrates that using Heat-ray to help configure a network reduces the number of identity snowball attacks by 96%. This is a measure of the number of machines (out of 1,000) that are reachable and can be compromised before applying Heat-ray and after multiple iterations of Heat-ray.

2.3 Mobility

The authors of [32] examine user movement with cell phone records. They suggest that user predictability follows a fat-tailed distribution. This means that users that travel less should be easier to predict and those that travel farther are less predictable. Said differently, the entropy is higher for those that regularly travel farther. They also point out that there is a threshold (they find to be 10 km) in which all users after this point are about equally predictable, although less predictable than lower distances. They indicate that there is a potential 93% average predictability in user mobility. They do not find any factors such as demographic, age, gender, or even weekend to weekday comparisons to be factors in predictability.

The authors of [33] look at wireless access point data gathered during an ACM conference. They have data from four APs located in each corner of an auditorium. They are able to correlate data with the events of the conference schedule. With respect to user mobility, they determine the number of access points visited and the number of access point handoffs that occur. They notice fewer APs visited on the half day of events when compared to the days that had full schedules, indicating less user roaming. The number of AP handoffs over time also indicates points throughout the day when users were not moving and started moving, which they correlated with breaks between conference talks.

Others have done characterization of larger wireless networks. [25] looks at 476 access points spread across 161 buildings, but only identifies 1706 unique wireless users (unique MAC addresses). They gather data using syslog, SNMP polling, and tcpdump. In 2008, the "largest WLAN study to date" [27] examined 7000 users across 550 access points. This study also uses syslog, SNMP polling, and tcpdump captures, as well as telephone (VoIP) records. They obtain 32,747,757 syslog messages, 16,868,747 SNMP polls, and 4.6 TB of sniffed traffic.

Most of the prior work that uses wireless access point traffic either needs data that must be gathered on demand for desired experiments (additional logging software, hardware, sniffers, etc.) or configuring access points to save copies of packet traffic. Network packets are less descriptive than event logs at an application level, although may contain other useful lower level information. For example, we might not be able to tell that the packet is a

TGT request, but we can instead know the client IP/MAC addresses, access point IP/MAC addresses, and signal-to-noise ratio.

We also notice that results seem in conflict across different research studies. [27] and [26], for example, see much different mobility patterns. Henderson et al. see most of the users spending a vast majority of time staying at the same access point, while Balazinska and Castro see more mobile users. Interestingly, the campus environment was observed to have less mobile users compared to the corporate environment. Mobility patterns are likely to vary for differing campuses (e.g., business vs. college) but past work indicates the patterns may not be obvious or intuitive.

We investigate an alternative approach that takes advantage of existing logging architecture. Windows event logs are generated at Active Directory controllers when users request Kerberos TGTs, request Kerberos service tickets, or log into wireless access points, among other triggers. This is a common procedure for IT departments for security purposes. These logs are already being gathered for the purpose of security auditing, so we attempt to reuse them to answer mobility questions.

Across mobility papers, we find that the common areas of interest are:

1. Number of users/connections over time
2. Average number of users at an access point over time
3. Amount of data transferred

We are not working directly with access point logs or packet captures, unlike these papers. We must translate these important considerations into the paradigm we are working with:

1. Number of users/connections over time
2. Average number of users at a domain controller over time
3. What events were logged and how many of each event

CHAPTER 3

EXPERIMENT

3.1 Proposal

Imagine a situation exists in which an attacker has gained access to one computer on a network through some vulnerability (e.g., code injection attack) or user error (e.g., malicious e-mail attachment). If the attacker wants to expand his access, he may wish to gain access to other computers or resources in the network. This is also known as network lateral movement. For example, the attacker may be able to compromise a local account on a regular workstation. That account might not have gone through the Kerberos protocol and gained access to certain resources (i.e., possessing service tickets). Alternatively, the attacker may want to log into a particular resource using a particular account (e.g., access a network folder as the user "Dan Smith"). There are multiple attacks that can be performed to move laterally in a network.

We are building models of user behavior on a campus network. Before we can model lateral movement in the context of what an attacker is capable of, we first look at what we should expect from regular movement.

Active Directory controllers, TGSs, and Kerberos services all log network events. These logs are sent to an aggregation point where they can be collected, stored, and later analyzed. In this study, we look at event logs generated from Active Directory controllers. In the future, we want to also look at TGS and Kerberos service logs.

3.1.1 Collaboration

We collaborated with Technology Services at the University of Illinois Urbana-Champaign to obtain Active Directory data logs. Technology Services ad-

ministers Active Directory domain controllers and some services that use Kerberos authentication.

3.1.2 Institutional Review Board

This study involved collecting data of human subjects, so it was our obligation to take every necessary precaution to ensure subject privacy and ethical data collection. We submitted a New Protocol Application to the University Institutional Review Board (IRB). This "Application for Review of Research Involving Human Subjects" described the study in detail and outlined the precautions we took to ensure responsible and ethical collection, handling, and storage of user data. These precautions are described next.

3.2 Ethics

3.2.1 Privacy Safeguards

Logs gathered contain account names (NetID), host names, client computer names, client network addresses, Kerberos session information (event codes, error codes, encryption type), and timestamps.

Usernames, client computer names, and potentially client network addresses are user-identifying information and were anonymized. These fields were replaced with a number chosen from the space of all numbers such that for each username, its corresponding random number will always be the same. Additionally, we could not map any corresponding random number back to its original data without a key that is held solely by Technology Services. To be more specific, we used a keyed hash algorithm that is constructed from the 256-bit Secure Hash Algorithm (HMAC-SHA256) [34]. In other words, username data is unique so that we can differentiate between users, but it is not possible to identify a username based on a hashed username. Further, we did not circumvent these protections by attempting to re-identify the users. The key is held by Technology Services, meaning we did not have the ability to de-anonymize users.

The anonymized data was periodically uploaded to an aggregation point. The aggregation point is a Technology Services-administered shared Box di-

rectory. Box is a service for sharing files, and is approved by the university of storage of FERPA sensitive data [35]. This was where all the logs were collected and combined. We then synchronized data from the shared Box directory with our Network and Security Research Group (NSRG) server. The data transfer took place over HTTPS. This provided an encrypted, secure channel of communication for the anonymized data. The NSRG server did not communicate with the workstations, AD controllers, or services.

The NSRG server is located in the Advanced Computation Building (ACB). Data resides on a Virtualized Machine running on the server. Data handling risks were severely curtailed through the use of best practices in securing the collection infrastructure and processing machines. These include, but are not limited to: locked office, restricted access, restrictions on copying study-related materials, access rights terminated when authorized users leave the project or unit, individual ID plus password protection, encryption of digital data, network restrictions, no non-UM devices used to access project data, security software (firewall, anti-virus, anti-intrusion) installed and regularly updated on all servers, workstations, laptops, and other devices used in the project. All data storage and processing occurred on the NSRG server, and the anonymized data did not leave the server, except in aggregated form for research presentation.

The participants accessed Technology Services-administered computers and performed their intended tasks as normal. This includes, but is not limited to, working on homework, writing papers, programming, using network printers, checking personal and university emails, web browsing, and playing games. We did not interfere with participants' computer usage and our data gathering was transparent to the user. This is identical to how Technology Services currently gathers data about user activities for network security purposes.

3.2.2 Risk Analysis

We believe the users would experience minimal distress if they discovered that their usage was monitored. We believe this because Technology Services-administered computers are identified as systems that are being actively monitored for analytics and security purposes. The collection process was transparent to the users and did not cause undue stress on their computing

needs.

We believed the risks involved were minimal. The data gathered was anonymized and stored on a secure NSRG server. If it were to be leaked, individuals and personal information would not be revealed. We did not expect participants to feel any additional psychological stress that they would not otherwise undergo from standard IT data logging.

We believed that the potential benefits of this research were significant. We were able to quantify the opportunities an attacker has while attempting to move laterally in a network. This was determined by identifying how many resources an attacker would have available that can be used to gain access to additional network resources. It provides cyber security researchers and Information Technology personnel with insight on how attackers can navigate a compromised network and what network resources can be targeted. These insights could help in threat mitigation, recovering from compromises, and identifying if an attacker is moving around in an internal network. These threats directly impact organizations, corporations, universities, and other entities that use internal networks.

3.2.3 Log Contents

We needed to know the exact contents of the logs we would be analyzing. We also needed to determine which fields in the log contained revealing information that must be anonymized. We received a sample log from Technology Services. This log contained information from only our NSRG lab volunteers. Details about the log content, including descriptions of each field and event code translations, can be found in Appendix A.

3.3 Process

3.3.1 Data Gathering Process

Data was collected from Technology Services Active Directory (AD) controllers and workstations. Data collection tools (i.e., AD service logging) are running on the AD controllers and workstations that collect data and store them in logs. Details about all the log fields are provided in Table A.1.

Usernames, client computer names, and potentially client network addresses are user-identifying information and were anonymized by being replaced with an HMAC-SHA256 hash [34]. To reiterate, username data was unique so that we can differentiate between users, but it was not possible to identify a username based on a hashed username.

The anonymized data was periodically uploaded to the NSRG server through the process previously described. The data was uploaded to a shared Box directory, and downloaded from Box to the NSRG server.

3.3.2 Anonymization Pre-Processing

The logs generated will contain sensitive information, which means that anonymization must take place before we receive the data. Anonymization must occur at a Technology Services computer prior to being transmitted to the NSRG server. We took this into consideration when developing the anonymization technique. We wanted to minimize inconvenience and manual labor, and maximize the data acquired. We wrote scripts to anonymize sensitive fields, which would be used in an automated process. This reduces inconvenience and manual labor. We also wanted to minimize inconvenience by not imposing any unrealistic requirements on Technology Services to run our script.

The script reads in the CSV log, anonymizes sensitive fields, and saves the result. As mentioned previously, the fields we consider sensitive are those that contain information that could be considered identifying. In these logs, we anonymize fields that contain NetIDs, IP addresses, and device names. Fields that contain only a NetID or IP address, such as "Account_Name," "Logon_Account," and "Client_Address," are anonymized using the HMAC-SHA256 keyed hash function. We anonymize the field "Source_Workstation" more tactically. This field may contain information about the source device that generated the log, such as a MacBook Pro or iPhone. It also may contain identifying information about the owner of the device. For example, our sample logs contained the entry "Zanes-MacBook-Pro-2.local" which identifies one of our volunteer's devices but also his first name. We used the sample logs to identify patterns to look for when parsing the logs. In the anonymization script, we used regular expressions (regex) to match

these patterns. When a pattern was matched, we stripped any information that could be considered sensitive and left only the pattern. For example, "Zanes-MacBook-Pro-2.local" would become "MacBook-Pro." This removes user privacy concerns, but allows us to gather statistics about what devices are used on the network. We took the safest approach if a pattern was not matched. If we could not identify the contents of the Source_Workstation field, we HMAC-SHA256 hashed the entire field. We chose to do this to ensure no sensitive information would be revealed in the case that we encounter a device that we did not account for. This field is also self-reporting, according to Technology Services. Therefore, we were conservative with this field because it could contain anything the computer or user chose to label itself as.

Our first attempt used PowerShell 5.1. We chose PowerShell because it is installed on Windows computers by default, which is what Technology Services uses. Therefore, no additional installations or setup were necessary. We discovered, however, that PowerShell was not practical for pre-processing data fast enough or at the scale we were working with. In a test environment using a 700 MB log sample, the PowerShell script took 1 minute 53 seconds to read the file into memory, 1 minute 35 seconds to perform anonymization, and 55 seconds to write the final data to disk. The script memory consumption was also not practical for data logs of the size we expected. PowerShell used approximately 10 GB of RAM to import the 700 MB log file into memory. We speculated the memory usage was much more than the size of the file because the PowerShell "import-csv" command-let generates a dictionary-like data structure. We predict PowerShell uses a substantial amount of memory for CSV metadata. This was with a 700 MB file, but the practical performance would be worse than this because the actual logs were many gigabytes in size.

We gave the PowerShell script to Technology Services to test performance and verify functionality. They ran the script on two log samples. The first was a sample over a 5-minute period and was about 118 MB. The anonymization script completed in about 4 minutes 22 seconds. The read, anonymize, and write functionality was timed, as well. Importing the CSV took about 1 minute 35 seconds, anonymization took about 2 minutes 32 seconds, and writing the anonymized data took about 15 seconds.

The second sample was over a 30-minute period and was about 711 MB.

PowerShell was processing this log for over 22 hours before being cancelled. Importing the CSV took about 8 hours 47 minutes 3 seconds. It was in the process of anonymizing before being cancelled.

We created multiple variants of the script which incorporated parallelization, reading the log as a stream, and reading the log in chunks. All variants had worse performance than the first version.

We used the PowerShell script as a template and wrote a Python 2.7 version of the anonymization script. Running the first draft, unoptimized Python script on the previous 700 MB log sample resulted in a start-to-finish time of about 90 seconds. The substantial difference between this result and the PowerShell results caused us to re-evaluate the anonymization approach. Some amount of setup or installation would be worth the gained performance benefits from using an alternative scripting language.

Table 3.1 displays the start-to-finish run times of some of the scripts we created. All the run times shown are for a 3 MB log sample. PowerShell Standard reads the entire CSV into memory using the "import-csv" command-let, performs anonymization to the data in memory, and then writes the data back using the "export-csv" command-let. PowerShell Stream-Read reads and anonymizes the file one line at a time, rather than importing the entire CSV into memory. It then writes back using the "export-csv" command-let. PowerShell Batched and Parallelized X reads in X lines of the file, starts a new thread to perform anonymization on those X lines, and then writes the anonymized data back as a new, smaller CSV. Python Read Then Write reads and anonymizes the file one line at a time, keeping the contents in memory, and then writes back to disk. Python Read-Anonymize-Write-Repeat reads and anonymizes the file one line at a time. After it reaches a threshold of 10,000 lines, it writes the anonymized lines to disk. It therefore only keeps 10,000 lines worth of log contents in memory at a time. The significant performance difference we observed between PowerShell and Python, both in Technology Service's test run and our own testing, motivated us to switch to Python.

Table 3.1: Script Run-Times

Script Description	Run-Time (seconds)
PowerShell	1.178
PowerShell Batched and Parallelized 500	4.291
PowerShell Batched and Parallelized 1,000	3.665
PowerShell Batched and Parallelized 10,000	3.920
PowerShell Batched and Parallelized 100,000	3.893
Python Read Then Write	0.440
Python Read-Anonymize-Write-Repeat	0.188

3.3.3 Transferring Data

Once the data has gone through anonymization pre-processing, it is ready to be sent to the NSRG server. The anonymized data will be periodically uploaded to a Technology Services-administered shared Box directory. This is where all the anonymized logs will be collected and stored. We then synchronize data from the shared Box directory with our Network and Security Research Group (NSRG) server.

We wrote a Python script to automatically copy files from the shared Box directory to the NSRG server. The script uses the official Box Python SDK [36]. This uses the Box API to authenticate, copy, and delete. The script authenticates as a user client and copies all files from the shared Box directory to the NSRG server. When the files copies are complete, we then delete the copied files from the Box directory. We do this to save space in Box (a log over an eight hour period can be 3 GB or more). These data transfers take place over HTTPS. Once on the NSRG server, the data is ready to be processed.

3.3.4 Data Processing

All post-anonymization processing was done on the NSRG server. The server has 128 GB of RAM and 32 logical processing cores. Statistics gathering and graph generation were done using Python 3, particularly the numpy, matplotlib, and scipy packages [37, 38, 39].

Every entry has a timestamp ("_time") and one of two name fields will be used. If an anonymized username is present, it will be in either the "Account_Name" field or the "Logon_Account" field. Which field has the

username depends on the type of event that is logged. We have observed that when a username is in the "Logon_Account" field, it is primarily for event code 4776. We will discuss event codes more later. This event states that "the domain controller attempted to validate the credentials for an account" [40]. All other events that use a username field have the username in the "Account_Name" field. Another event code of interest is 4768: "A Kerberos authentication ticket (TGT) was requested" [40]. Two more event codes that will be important when looking at user activity are 4634 ("An account was logged off") and 4624 ("An account was successfully logged on") [40].

The first iteration of the Python script was single-threaded with little consideration put into memory consumption. This worked fine on a 5-minute sample log we initially received from Technology Services. Once we received logs covering 24 hours of activity at a time, our script was no longer practical. The script was using all 128 GB of memory and disk swap memory was continuing to increase.

We had to optimize the script to handle the amount of data we were dealing with. The first iteration read every line of the CSV into a list. Each entry in the list was a dictionary. The dictionary keys were the field names (e.g., Account_Name, EventCode, etc.) and those mapped to the corresponding fields' values. This method was not feasible for even 24 hours of log data at a time, let alone a week's worth of data that we would later be handling. These logs for one day were about 25-30 GB in total.

We overcame this by parallelizing the Python script. The revamped script reads every log file in a given directory into memory simultaneously (up to the number of cores available). Instead of saving a list of dictionaries for all the data, we are more selective about the data we look at. With each simultaneous file read, we build dictionaries of only information we are interested in. For example, we want to look at the number of users we encounter in the logs. We go through each line of each CSV (in parallel) and make a dictionary of usernames. The usernames are the key and the value is the number of times this user was encountered.

3.4 Log Analysis

The anonymization script hashes usernames, but we want to ensure the same username results in the same hash even if the field is formatted differently. The script attempts to pull NetIDs out of username fields of various formats. We use regular expressions to match all possible formats from the sample set we obtained from Technology Services. For example, in the sample data of only our information, there are fields such as "CITES-IDM TDI user\nCN=<NETID>,OU=People,DC=ad,DC=uillinois,DC=edu" where <NETID> would be filled with a user NetID. We worked with Technology Services to identify all patterns we might encounter, but we cannot guarantee all of them are accounted for.

In the first iteration of data anonymization and analysis, we do not distinguish between users, services, and network shares. This was primarily because we did not see services and shares in our sample log. We were unaware that they would be included in the live anonymized logs, let alone their username patterns.

One of the first things we want to look at is usage patterns. We expect to see periods of time with little activity, such as early in the morning (midnight to 6am) before students and faculty arrive. We then expect to see an increase of activity in the morning as students are waking up and faculty arrive. We expect relatively steady activity throughout the day as students go to class, others get out of class, and students and faculty use the internet throughout the day. We expect a slight decline in the late afternoon or early evening as faculty leave work for the day. We suggest a slight change because the student population (about 44,000) is over eight times larger than the faculty and staff population (about 5,000) [24].

We also speculate that Monday and Wednesday will have similar patterns, and Tuesday and Thursday will look similar. Classes at UIUC are typically scheduled at the same time on these day pairs. Therefore, we predict that similar usage patterns will result from similar student class attendance patterns.

We look at log data covering a week of events from November 1st to November 6th. These logs were a total of about 220 GB.

3.4.1 Events

We are working with Windows event logs that are generated at Active Directory controllers. Each log entry has an event code describing what event occurred. We should therefore investigate what events we are encountering most. Table A.2 contains descriptions of each event code encountered. Table B.1 shows statistics about all event codes from all user activity over the entire data set. The top five event codes produced are 4624, 4634, 4776, and 4768. All of these events are related to a user logging in (or out). As a sanity check, we see that events 4624 and 4634 are roughly equivalent. The total number of account logins and log outs are about the same. From this we can say that top events we encounter are for user authentication. The event code descriptions for the top ten events sorted by total number of events are in Table 3.2.

We can identify from this table some of the answers we seek. First we look at TGT distribution. Event code 4768 is described as "a Kerberos authentication ticket (TGT) was requested." There were 46,717,178 total TGT request events and 6,830,640 unique user TGT request events throughout the week. This comes to about 6.8 TGTs per user if we assume a uniform distribution.

Event code 4769, "A Kerberos service ticket was requested," occurs 3,891,719 times. These events occur from 404 unique users. This comes out to about 9633 service tickets requested per user given a uniform distribution. We do not currently possess information in the logs that identifies the service, but it is intended for future work.

The next event, 4776, is described as "the domain controller attempted to validate the credentials for an account." From discussions with Technology Services, we know that these events occur as a result of an NTLM authentication. There were 95,899,306 total NTLM authentications and 4,640,444 unique user NTLM authentications. This becomes about 20.66 NTLM authentications per user assuming a uniform distribution.

We also see some event codes that correspond to administrative tasks, such as event 4672 and 5136. We take a closer look at these events later.

We want to look also at event codes produced from unique users. Perhaps users have misconfigured hardware or software. Maybe some are having connection or authentication problems. There could also be automated tasks

Table 3.2: Top 10 EventCode Descriptions Sorted by Total Connections

Event Code	Description
4624	An account was successfully logged on
4634	An account was logged off
4776	The domain controller attempted to validate the credentials for an account
4768	A Kerberos authentication ticket (TGT) was requested
4672	Special privileges assigned to new logon
4771	Kerberos pre-authentication failed
4648	A logon was attempted using explicit credentials
4769	A Kerberos service ticket was requested
5136	A directory service object was modified
4625	An account failed to log on

that rapidly log in and out. In an attempt to reduce these events, we look at event codes per user. In other words, a user account only contributes one count for any events they produced. The event code descriptions for the top ten events sorted by unique users are in Table 3.3.

We see many of the same events top this chart. Login (4624) and log off (4634) are about the same again. According to [40], log off events are not properly logged by Windows until the system restarts. They further say that a logoff event may not be recorded if there is an unexpected shutdown or loss of network connection. These seem to be likely explanations for the two events not being exactly the same.

When we compare the top contributors in Tables 3.2 and 3.3 (or the numbers in Appendix B.1), we see that NTLM authentication events are a larger percentage of the total events compared to the unique events. This indicates that accounts are issuing 4776 NTLMauthentication events more frequently than the other login/logoff events.

Table 3.4 shows how many instances of each event code were found without a username associated with the event. Event codes that do not have missing usernames are not listed. Over 31% of login and logout events are without a username. Almost 90% of "special privileges assigned to new logon" do not contain a username. About 65% of "a privileged service was called" are without a username, too. As we will see later, a subset of administrative tasks leave the username field empty.

Table 3.3: Top Ten EventCode Descriptions Sorted by Users

Event Code	Description
4768	A Kerberos authentication ticket (TGT) was requested
4624	An account was successfully logged on
4634	An account was logged off
4776	The domain controller attempted to validate the credentials for an account
4771	Kerberos pre-authentication failed
4648	A logon was attempted using explicit credentials
4756	A member was added to a security-enabled universal group
4732	A member was added to a security-enabled local group
4733	A member was removed from a security-enabled local group
4625	An account failed to log on

Table 3.4: Number of EventCodes that Appear with Empty String Username

Event Code	Number With Missing Username	Percent of Total
4624	33,730,337	31.21
4634	33,730,104	31.21
4672	33,422,666	89.46
4673	211,539	65.33
4648	4,702	0.036

3.4.2 Access Behavior

We next characterize user activity over time. We look at event log data covering a week's time. Figure 3.1 shows total user activity from midnight on November 1st to midnight November 7th. Activity is grouped into one hour bins. This total activity includes events that the same user may have produced within the bin. We see a pattern of lower activity in the early morning, increasing activity over the day that peaks around noon, followed by decreasing activity throughout the rest of the day. There are spikes of extremely high activity at about 2am every day. There is also an extreme spike at about 11am on 11-01. We analyze these more later.

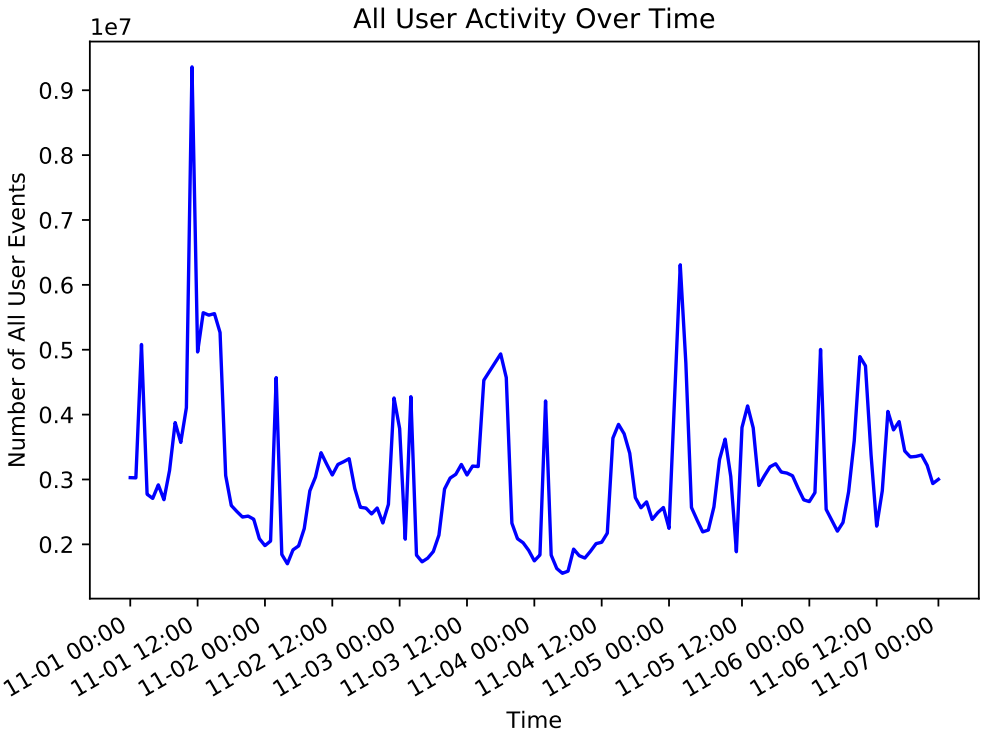

Figure 3.1: User Activity over Time

To avoid a few power-users from distorting the graph, we also look at unique activity. Figure 3.2 shows unique user activity over the same time period. Activity is again grouped into one hour bins. This unique activity includes events produced by unique individuals only. Even if the same user produced multiple events within the bin, it is counted as one event. We immediately see a difference in the magnitude of activity. Total non-unique

user events reached almost 10 million in one hour. The unique graph shows activity maxing out at no more than 120,000 events in an hour. This indicates that there are users producing many events per hour. The reoccurring 2am spikes are no longer obvious in this view. We can see defined peaks for each day. Wednesday, Thursday, Friday, and Monday have about the same magnitude. Saturday and Sunday clearly have fewer events being produced. Contrary to what we predicted, Monday and Wednesday do not have obvious shared patterns when compared to any other days.

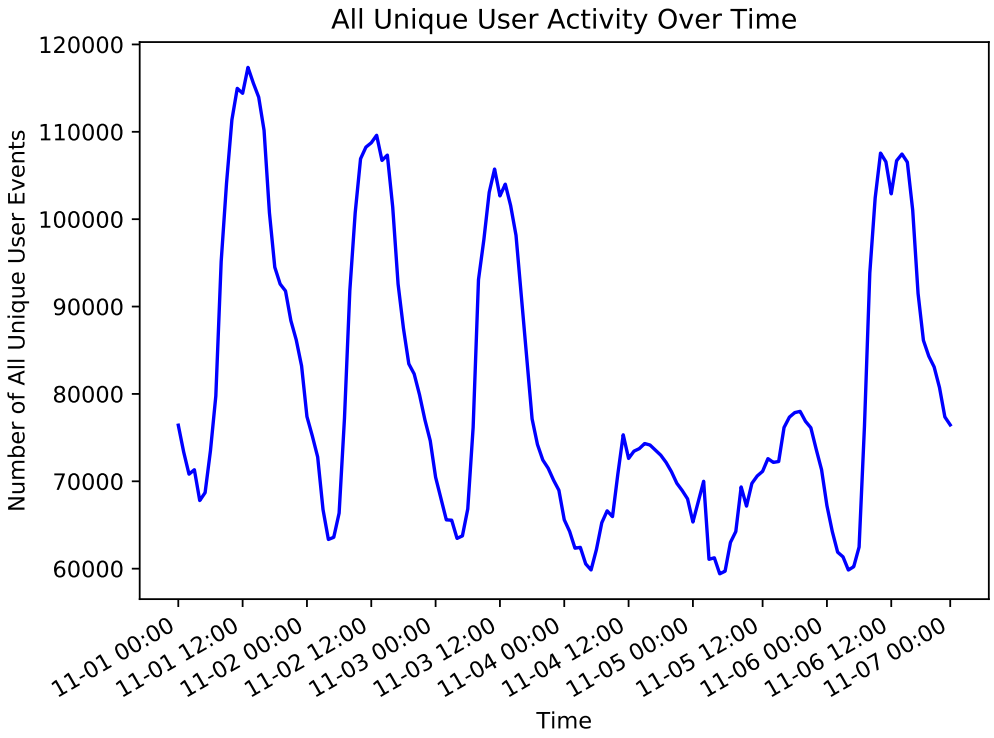

Figure 3.2: Unique User Activity over Time

Figure 3.3 shows the top five most seen event codes over time. Figure 3.4 shows the next five most frequent event codes over time. These are broken up to make it easier to read. Please refer back to Table 3.2 for the top ten event code descriptions. From these we can see what specific events causes the activity we previously saw.

Table C.1 in Appendix C lists the correlation coefficients calculated between each event code pair. We focus on the top ten events for identifying correlation.

We see from the pattern of 4768 that TGTs are requested at a steady

rate, increasing from early morning to mid-afternoon each day and gradually decreasing as the day goes on. Event 4776 (the domain controller attempted to validate the credentials for an account) occurs more frequently and in a more bursty pattern, but still following the trend of increasing to noon and decreasing after. Event 4776 activity is also lower on Saturday, then spiking to a weekly high Sunday after midnight.

The lines for 4634 (an account was logged off) and 4624 (an account was successfully logged on) are so similar that they are merged together on the graph. These two events have a correlation coefficient of 0.999. The one hour binning we did indicates that the logins and logouts are occurring within the same hour.

Event 4624 (logged in) and event 4776 (DC attempted to validate credentials) have a correlation coefficient of 0.586. Looking at Figure 3.3, we see that when one of them spikes, they spike together, but in non-spiking situations, their patterns do not mirror. Possibly a subset of logins are followed by credential validation. Since events 4624 and 4634 are so closely correlated, 4776 also has a correlation coefficient of 0.586 with event 4634.

Event 4776 (DC attempted to validate credentials) and event 4768 (TGT requested) have a correlation coefficient of 0.676. Event 4776 has a correlation coefficient of 0.628 with event 4769 (Kerberos service ticket was requested), which is shown in Figure 3.4. While they do not have the same magnitude, they follow the same trend of increasing from morning to mid-afternoon, followed by decreasing.

Events 4768 and 4769 have a correlation coefficient of 0.803. This again appears to be the situation where they have differing magnitudes (about 5 times magnitude difference) but similar temporal trends.

Events 4624, 4634, and 4672 (Special privileges assigned to new logon) have a correlation coefficient of 0.868. This event will not give us much information about user usage, according to Technology Services, as it is an administrative event. This can still imply that a portion of the login and logout events are related to these 4672 administrative events.

We investigate now the uncharacteristic spike that occurs in Figure 3.1 on 11-01 at about 11am. If we look at Figures 3.3 and 3.4, we see there is a correlation between events 4624, 4634, 4672, and 4776. There are about 2.6 million events occurring for each of 4624 and 4634 within the one hour. There are about 1.9 million events for 4672 in that same hour. There are also

about 1.4 million 4776 events that contribute to the spike. In addition, there are 1.9 million events during that hour that have the same "Source_Network_Address" (i.e., IP address). There are also 1.9 million events during the hour that originate from a single username. An additional 1.5 million events originate from a single, different username. This indicates that one or more administrative accounts are logging in, performing an action that is assigning a special privilege to a new logon, and then logging off. We say that this is an administrative account because only an account with escalated privilege can perform this event.

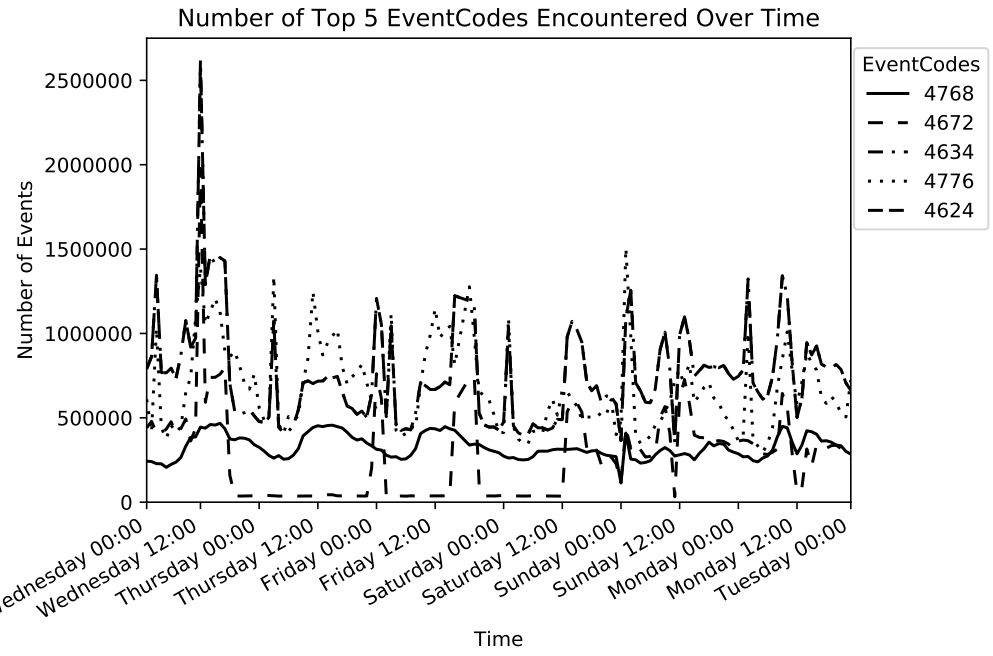

Figure 3.3: Top 5 Event Codes over Time

We next investigate the spikes occuring at 2am every day. Looking at Figure 3.4 reveals the event code primarily involved in the 2am spikes. Event 4648 (A logon was attempted using explicit credentials) has a small amount of consistent activity every day, along with one burst of activity at about 2am every day. This event could also be connected to the spikes we see in Figure 3.3 that also occur at about 2am. The spikes are less apparent in the latter graph due to the magnitudes of the other activities. Upon further inspection, we see the 2am bursts consist of events 4776, 4624, and 4648. The event code descriptions and repeated time indicate this is a scheduled, automated task. We return to investigate this reoccurring pattern shortly.

Event 4771 could be an interesting event for IT to follow. The description reads that a "Kerberos pre-authentication failed." There is a consistent level of these events that occur throughout the week. Some amount of these events above a certain threshold might be an indication of suspicious activity.

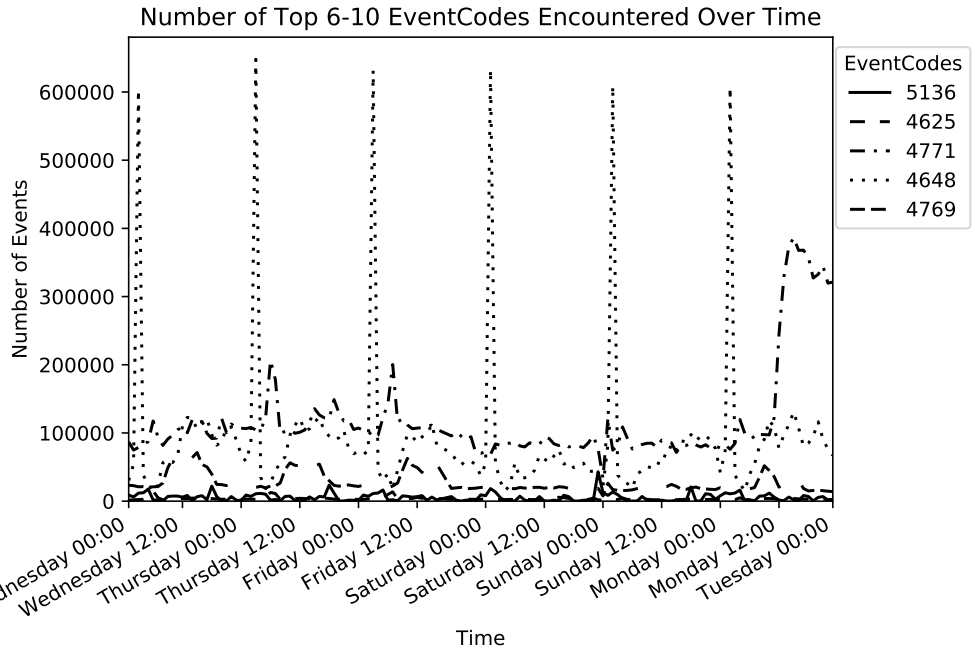

Figure 3.4: Top 6-10 Event Codes over Time

To make any daily repeating patterns clear, we now look at hourly usage graphs averaged over the week. This means taking the average of user activity per hour across each day. For example, 9:00 on the graph is the activity at 9:00 averaged across every day of the week. This is illustrated in Figure 3.5. The graph reiterates the daily patterns we previously saw, consisting of activity increasing from early morning to about noon, followed by decreasing activity. The average barely falls below 2 million events per hour at its lowest points. There is a high point early in the morning at about 2am, corresponding to the daily peaks we saw previously.

We look at the daily averaged events triggered by unique users once again. This is illustrated in Figure 3.6. In doing so, we see that the spike at 2am is gone. We speculated this could be some automated event that causes a sudden spike of network event activity. We reached out to Technology Services about this spike. They identified the source of the 2am activity to be a service account in an IT networking department. They suggested

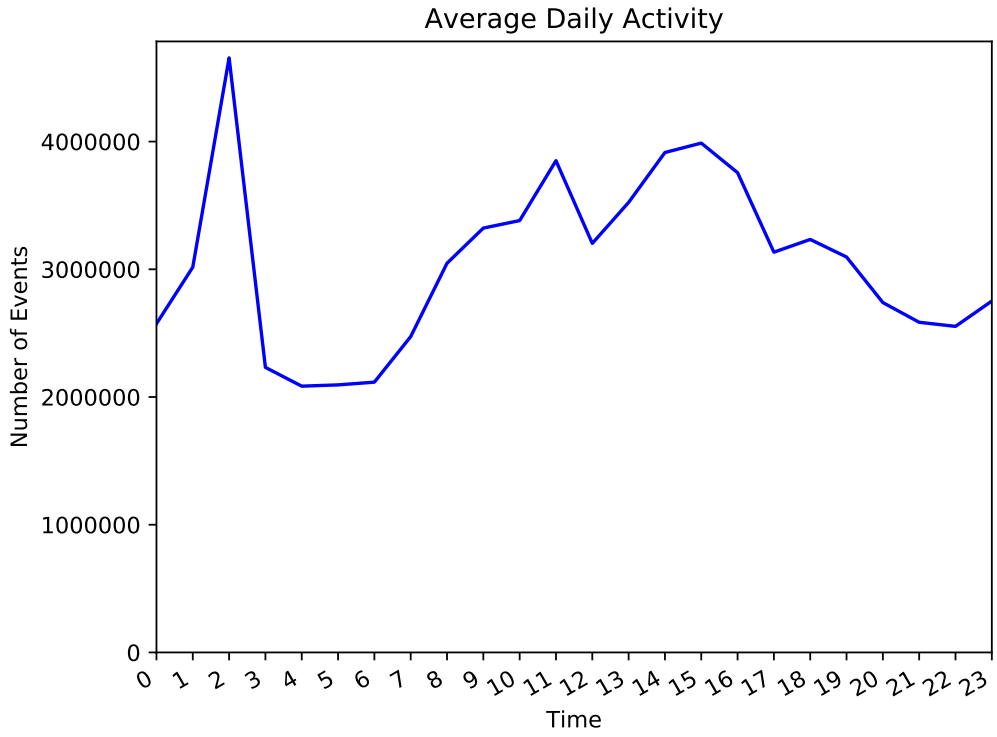

Figure 3.5: Average Daily Activity

that this could be "a nightly firmware check or log maintenance on all the networking switches on campus." Unique user activity per hour ranges from about 60,000 to over 90,000 events through the day. The unique login graph has been smoothed in comparison to the total login graph.

We found a total of 302,510 unique users. The maximum number of events created with the same user is 69,847,763. Upon further investigation, this is actually an empty string username. This means that these are all of the combined events for when a username is not present. This is most likely many users. The second highest number of events is produced by a non-empty string username. This account produced 11,151,510 events.

Next we look at the frequency of user activity. Figure 3.7 shows the CDF of user events over the week. On the x axis is a log-scale of the number of times a user creates an event. On the y axis is the frequency of events. The markers are every 10%. This tells us that about 5% of all users only create one event. About 50% of users create 100 events or less. About 70% of users create 1,000 events or less. Over 95% of users create 10,000 events or less.

Figure 3.8 shows the PDF of user events over the week as a scatter plot.

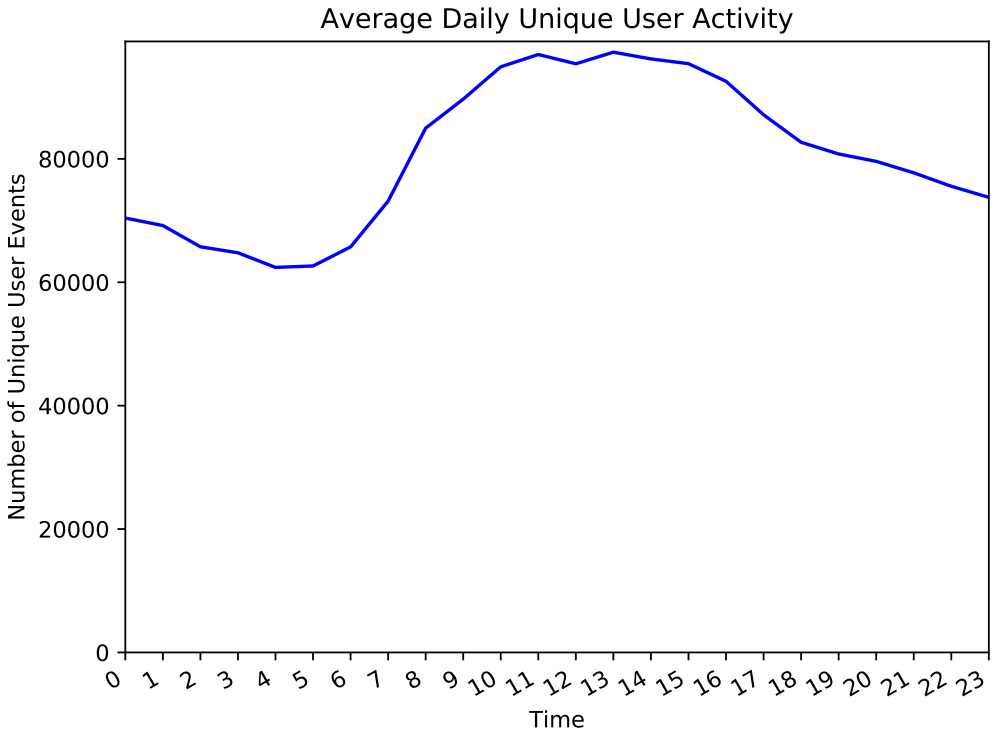

Figure 3.6: Average Unique Daily Activity

The y axis is the number of times a user performs an event. The x axis is the number of users that performed y number of events. Note that both axes are log scaled. This plot resembles an exponential decay in the number of events performed by users. The outliers are on the left end of the x axis. These individual users are the source of far more events than the rest of the users.

3.4.3 User Distribution across Domain Controllers

We next characterize user activity distributed across the campus domain controllers. We want to look at how many events are hitting each domain controller (also called a "host" in the logs) over time. Figure 3.9 shows the total number of events that target each domain controller over the week. Figure 3.10 illustrates the same but limited to unique users. Both graphs are binned into 24 hour periods.

There are two Amazon Web Services domain controllers (AWSDC), two Chicago Domain Controllers (CDC), three RADIUS servers, and six Urbana Domain Controllers (UDC). The CDCs serve infrastructure primarily

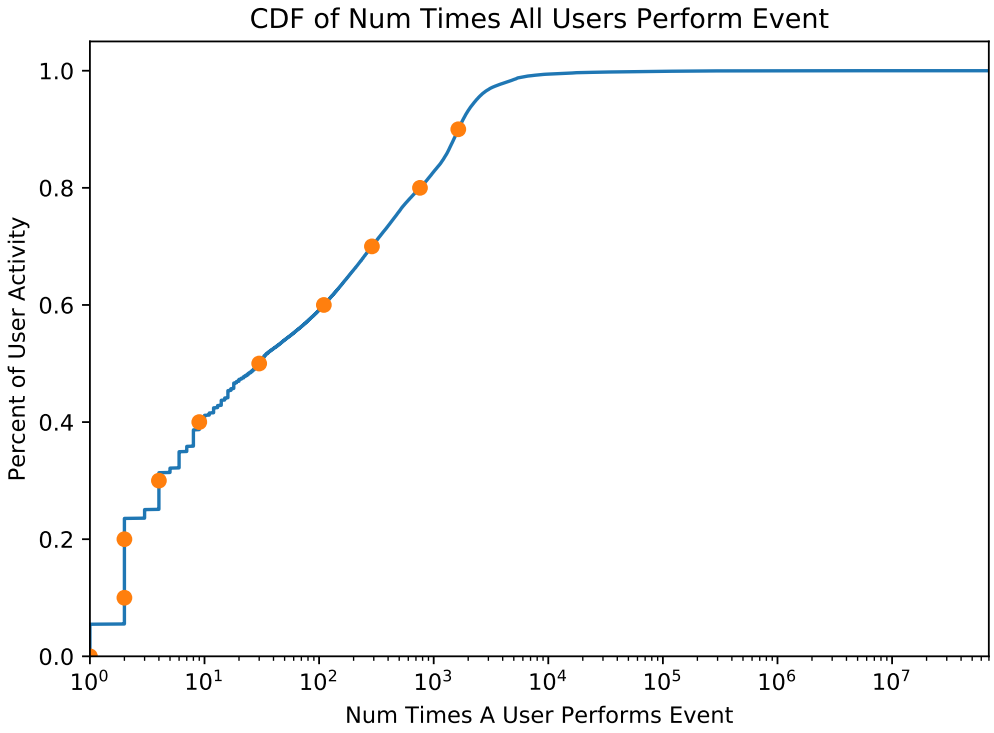

Figure 3.7: CDF of Number of Times All Users Login

in Chicago, where they are located, but not exclusively. They can still be reached from Urbana if a user explicitly tries to. The AWS DCs are for university AWS resources to use rather than connecting to a DC on campus. The Radius servers are used exclusively for Radius authentication, which are VPN connections and IllinoisNet WiFi connections. Of the three Radius servers, two are virtual machines (VMs) running on VMWare infrastructure in Urbana, and the third is a VM running on infrastructure in Chicago. All other traffic is served to the UDCs, located across the Urbana UIUC campus.

The majority of all traffic we see is going through the UDCs each day. When considering only unique users, we see the same usage primarily through the UDCs. When we compare the total and the unique, we notice that total AWSDC and CDC traffic appears to be originating from fewer individuals. This is because there are fewer unique user events and a larger number of total events for these two DC groups.

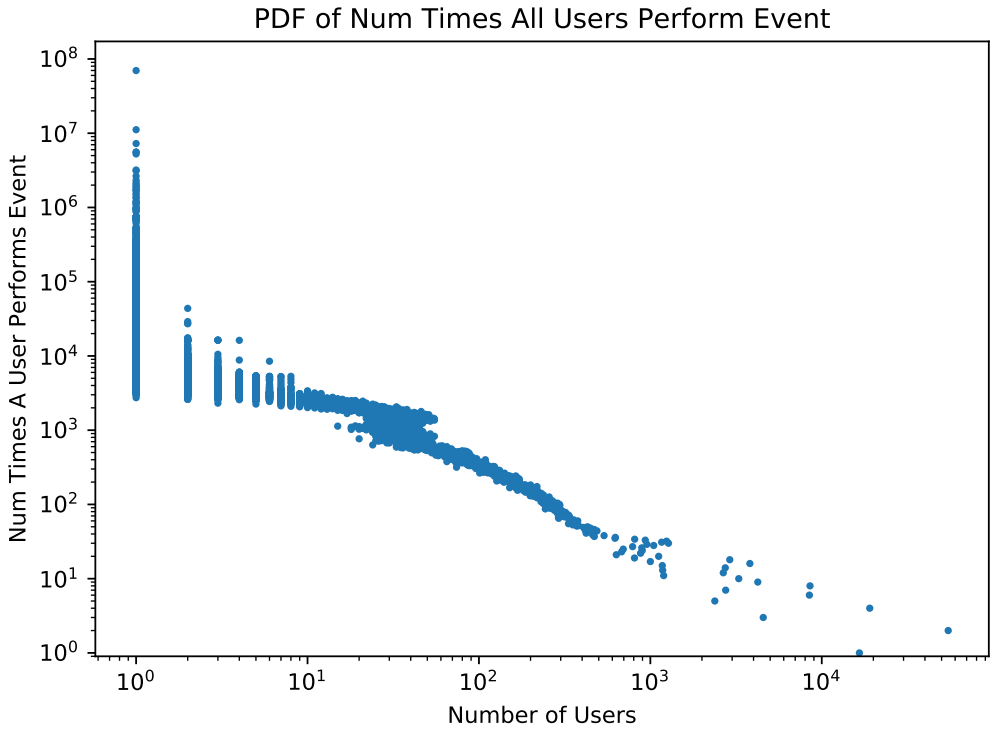

Figure 3.8: PDF of Number of Times All Users Login

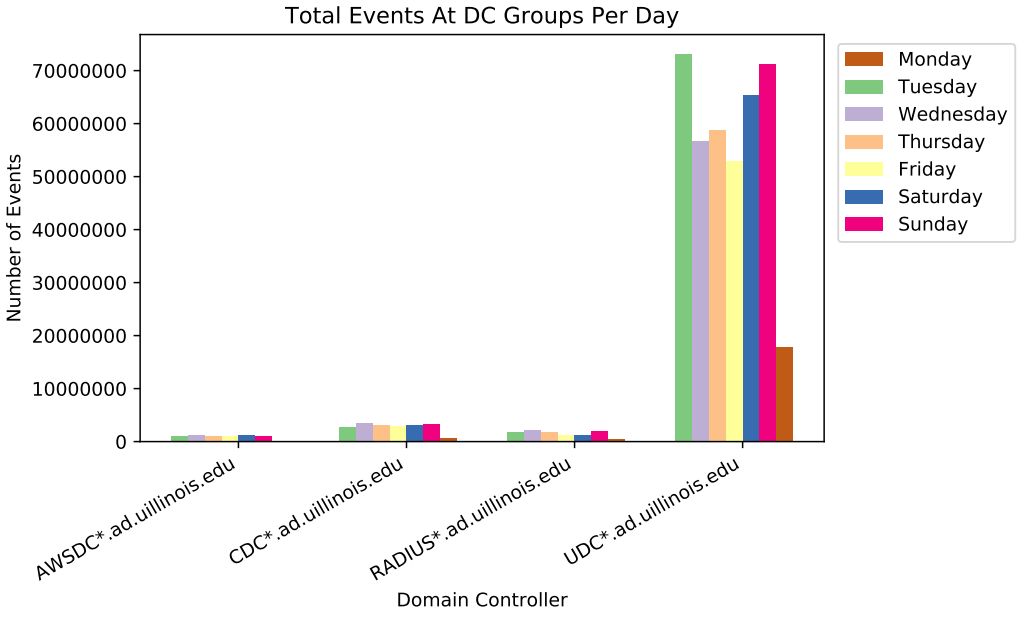

Figure 3.9: Total Number of Events at Each DC per Day

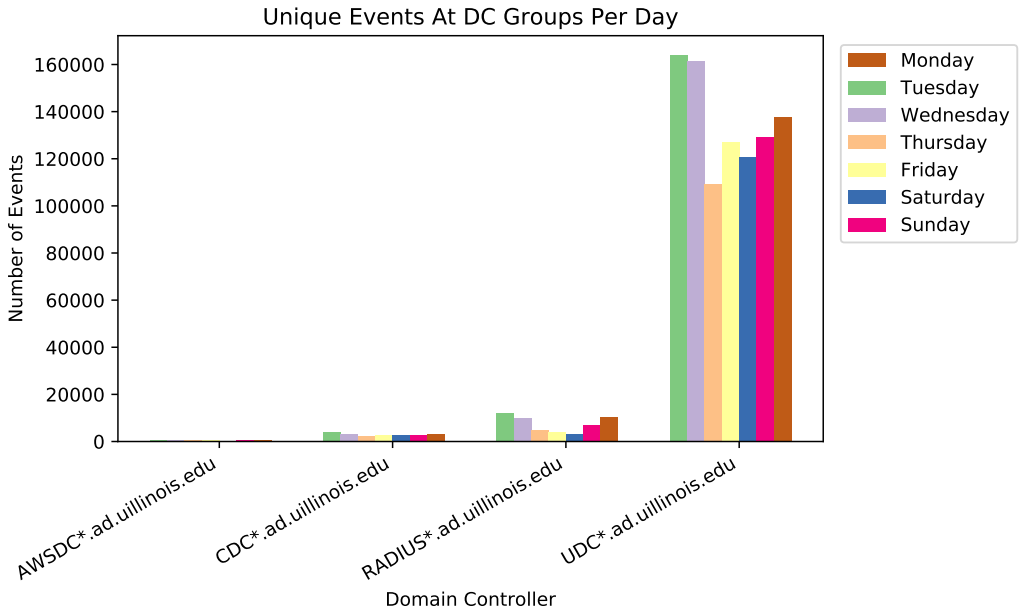

Figure 3.10: Number of Events From Unique Users at Each DC per Day

3.4.4 Filtering the Data

After discovering the source of the 2am spikes, we requested non-user-identifying information from Technology Services about the accounts that are the source of most events. They investigated between 11-01 and 11-06, and gave us descriptions of the top 20 users based on events produced. Table 3.5 shows these sources. We note that because of the anonymization process, we do not see the numbers match up exactly. We believe this is because of the grouping that occurs while anonymizing. That is, we might have fewer events linked to individual users because the usernames may not have been pulled out of the username field correctly. This is because we cannot guarantee we know all username patterns that exist. Technology Services knows the exact user that produced each event.

We now look at a second iteration of data anonymization and analysis. This time, we filter out services and network shares. Technology Services informed us that shares always have a dollar sign ($) and services are supposed to have a hyphen (-) in the name. We modify our regexes to account for these. We check if the username field is a share that includes a username. For example, “UDC02$\n<NETID>” where <NETID> is the NetID of a user that might be logging in and authenticating with domain controller UDC02. If it has the format of only a share with no username (e.g., “UDC02$”), we

Table 3.5: Top 20 Sources of Events Over the Week

Rank	Number of Events	Source Description
1	36,653,169	DC
2	34,412,744	DC
3	26,896,567	DC
4	17,175,232	Service
5	16,645,652	DC
6	10,887,342	Service
7	7,664,897	DC
8	5,392,531	Service
9	4,004,513	Service
10	3,705,068	DC
11	3,367,226	Staff
12	2,003,759	DC
13	1,918,936	Staff
14	1,884,371	Computer
15	1,785,325	Service
16	1,752,277	Student
17	1,707,985	Student
18	1,652,927	Service
19	1,613,793	DC
20	1,533,342	Computer

do not keep this event entry. If we find a hyphen in the username field, we do not keep this event entry, either. These efforts were additional attempts to filter out events that are automated or do not correctly represent a user-triggered action. We are now looking at information which we believe is more representative of user activity only.

We expect to see a smaller magnitude of events. If the reoccurring spikes are caused by service and/or share accounts, we also expect to see less spikes in the new set.

We first look at the event codes to get an idea of if or how the data we are about to examine might look differently. Table B.2 contains the event codes, total number of each event, total users that generated the event, and connections per user for events in the logs after filtering. The top five event codes, which are reiterated in Table 3.6, are now events that are exclusive to user login behavior. We no longer see log off events, however. Event 4624 has been filtered out entirely. This indicates that the source, or sources, of log off events were services, shares, or both. From what we saw in Table 3.4, 4624 accounted for 31.21% of empty string account names, as well. This means we should expect to see a decline of at least 33,730,337 empty string usernames, and 108,078,498 fewer overall events.

We also see that event 4648 (logon was attempted using explicit credentials) is no longer encountered in the logs. This suggests that all 4648 events were triggered by service or share accounts.

Events 4733, 4732, 4756, 4674, and 4757 in the top ten reveal that we have not filtered out all administrative events. Since we have filtered out services and shares, this indicates that the sources of these administrative tasks are non-service and non-share accounts. This does not exclude the possibility that the generation of these events is automated.

We find there are 121,834 unique users found in the week of time. This is based on the two username fields only. The pre-filtered logs contained 302,510 unique users over the week, making a difference of 180,676 usernames. Table 3.7 contains the event codes that that contain empty-string usernames and how many are encountered in the logs over the week.

In the filtered data, we see 35,507,152 TGTs were requested from 90,913 unique users, resulting in about 390.6 TGT requests per user with a uniform distribution. This is 11,210,026 less total requests than before filtering.

In addition, there are 78,763,663 attempts to validate credentials (NTLM)

Table 3.6: Top Ten EventCode Descriptions after Filtering

Event Code	Description
4776	The domain controller attempted to validate the credentials for an account
4768	A Kerberos authentication ticket (TGT) was requested
4634	An account was logged off
4771	Kerberos pre-authentication failed
4769	A Kerberos service ticket was requested
4733	A member was removed from a security-enabled local group
4732	A member was added to a security-enabled local group
4756	A member was added to a security-enabled universal group
4674	An operation was attempted on a privileged object
4757	A member was removed from a security-enabled universal group

coming from 91,315 unique users. This is 862.5 events per user with a uniform distribution. This is a loss of 17,135,643 in terms of total NTLM authentication attempts.

In the filtered data, we see that Kerberos service tickets were requested 2,748,806 times from 288 unique users. This comes out to about 9544.4 service tickets requested per user given a uniform distribution. This is 1,142,913 less total service tickets than the previous data set.

The maximum number of events from a single user in the filtered results is 11,296,413. This is the same username that we found to have created the most events in the non-filtered data, as well. Note that the number of events produced by the same user this time is more than previously (a difference of 144,903). This could be an indication that the update to our anonymization script is working as intended for this purpose. It may have previously not credited this user with events the user actually produced, and now is. This would explain the increase in events between scripts.

Notice also that the empty string is not the number one contributor anymore. As we see in Table 3.7, we are still observing events that have empty string usernames. The amount encountered is now fewer.

Table 3.7: Number of EventCodes that Appear with Empty String Username in the Filtered Logs

Event Code	Number With Missing Username	Percent of Total
4769	2299135	83.64
4776	108414	0.14
4768	20	0.00
4954	14	100.00
4946	10	100.00
4948	10	100.00
4755	1	0.13
4720	1	20.00
5141	1	100.00

3.4.5 Access Behavior

Figure 3.11 shows total user activity from midnight on November 1st to midnight November 7th. Activity is again grouped into one hour bins. We see a pattern of lower activity in the early morning, increasing activity over the day that peaks around noon, and decreasing activity throughout the rest of the day. We continue to see the repeated 2am spikes. The magnitudes of all events are less than before. The daily noontime peaks are about 1.8 million events now, while we previously saw noontime peaks of about 5 million. +

Figure 3.12 shows unique user activity over the same time period. Activity is again grouped into one-hour bins. This follows the same trend as the non-filtered graph, with reduced magnitude of events. We also see the 2am spikes are no longer obvious in this view. We can see defined peaks for each day, with the weekends having less activity.

Figure 3.13 shows the top five most seen event codes over time. Figure 3.14 shows the next five most frequent event codes over time. As mentioned previously, the top five events are all related to account login activities. Recall that event 4624 (logged on) has been filtered and so is no longer present.

Table C.2 in Appendix C lists the correlation coefficients calculated between each event code pair after filtering. We focus on the top ten events for identifying correlation.

Event 4776 (DC attempted to validate credentials) and event 4768 (TGT requested) have a correlation coefficient of 0.749.

Events 4768 and 4769 have a correlation coefficient of 0.798. This again

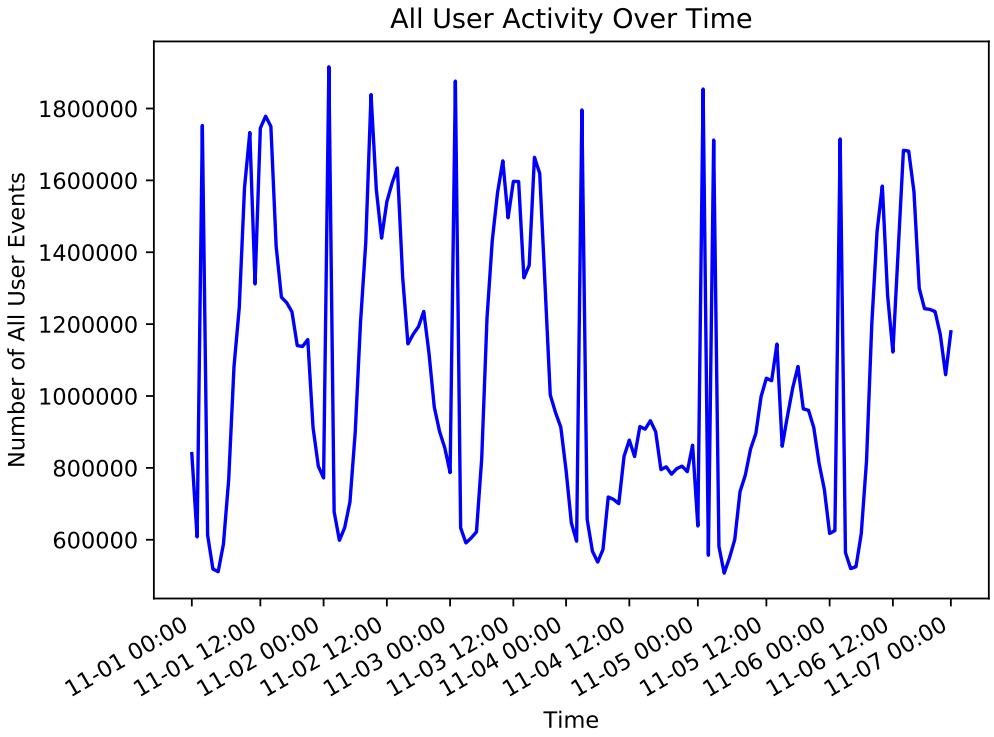

Figure 3.11: User Activity over Time after Filter

appears to be the situation where they have differing magnitudes (about 3 times magnitude difference) but similar temporal trends.

Event 4769 has a correlation coefficient of 0.6735 with event 4776.

Events 4634 and 4776 have a correlation coefficient of 0.719. This may indicate that a subset of credential validation and log off events occur in the same hour time window.

There is a correlation coefficient of 0.665 between events 4768 and 4771 (Kerberos pre-authentication failed).

Events 4732 (A member was added to a security-enabled local group) and 4733 (A member was removed from a security-enabled local group) have a correlation coefficient of 0.999.

Events 4732 and 4757 (A member was removed from a security-enabled universal group) have a correlation coefficient of 0.657.

We see a seemingly uncharacteristic spike of event 4771 (pre-auth failure) that occurs on Monday afternoon. When asked about this, Technology Services noted that the spike was caused primarily by one staff user on one computer. They note that this was one of the users that also showed up in

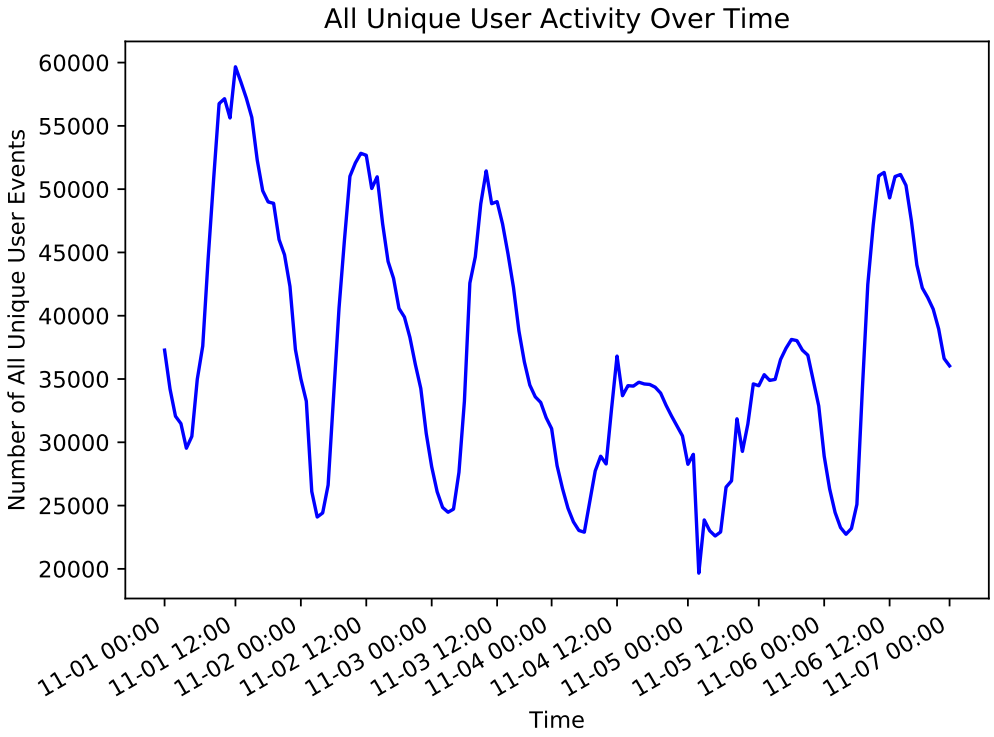

Figure 3.12: Unique User Activity over Time after Filter

the top 20 list.

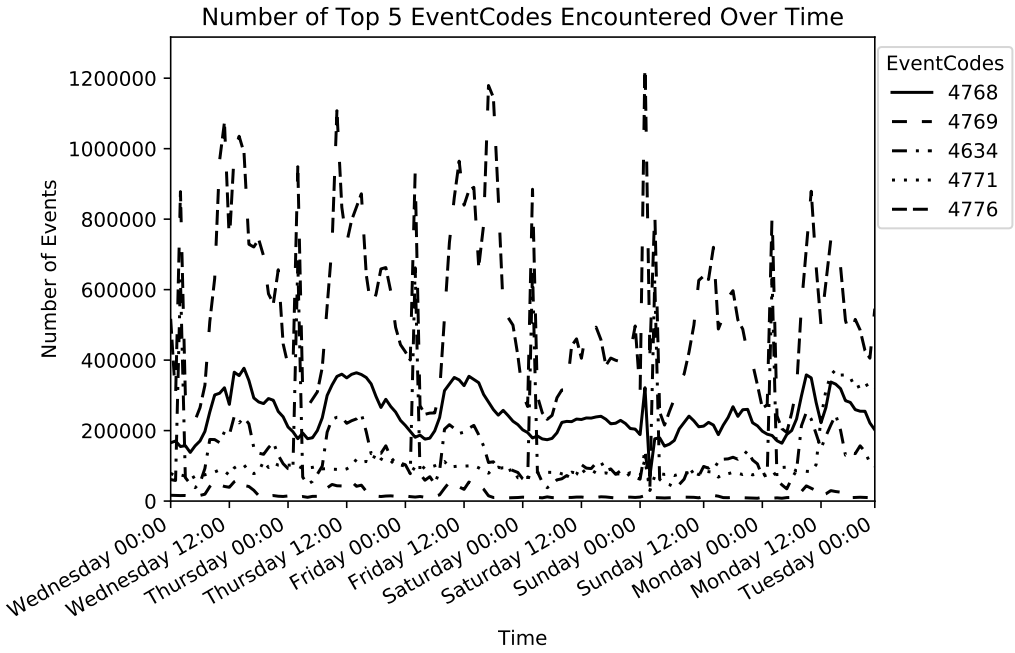

Figure 3.13: Top 5 Event Codes over Time after Filter

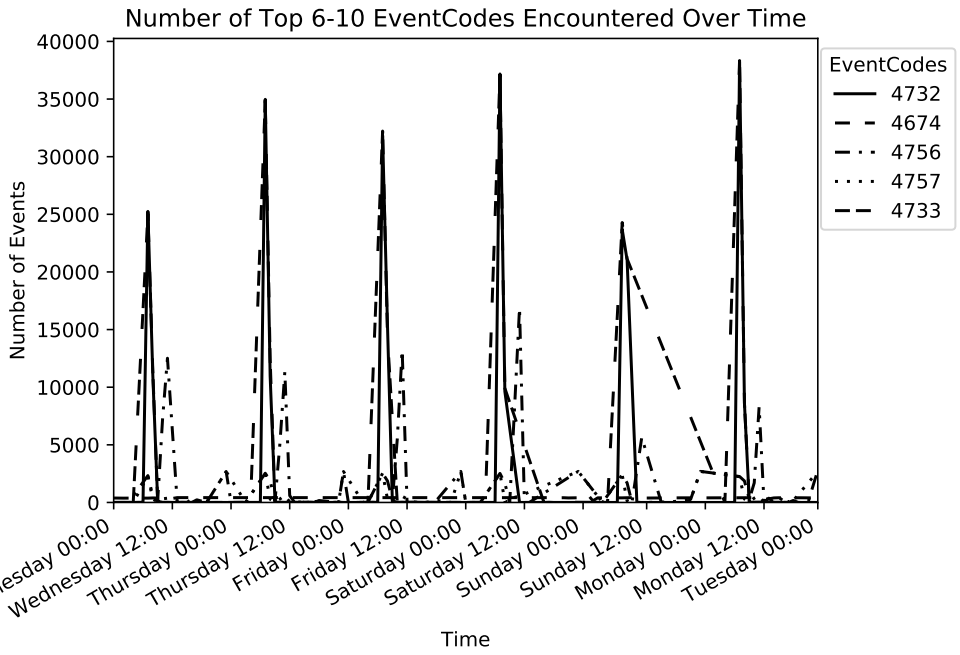

Figure 3.14: Top 6-10 Event Codes over Time after Filter

We now look at hourly usage graphs averaged over the week, illustrated in Figure 3.15. The average barely falls below 600,000 events per hour at its lowest points. There is a high point early in the morning at about 2am, corresponding again to the daily peaks we saw previously. We see a trend of activity increasing from early morning to about noon, followed by decreasing activity.

The average daily unique user activity can be found in Figure 3.16. We see a drop in magnitude of events, from the 500,000-2,000,000 range to the 25,000-50,000 range. The same users are responsible for multiple events per hour. This graph makes the trend more apparent: lower in the morning leading to an increase towards noon, and a slow decline as the afternoon goes on.

Next we look at the frequency of user activity. Figure 3.17 shows the CDF of user events over the week. On the x axis is a log-scale of the number of times a user logs in. On the y axis is the frequency of logins. The markers are every 10%. This tells us that about 5% of all users only create one event. About 20% of users create 10 events or less. About 50% of users create 100 events or less. Over 85% of users create 1,000 events or less. About 99% of users create 10,000 or less. The max number of logins from a single user was

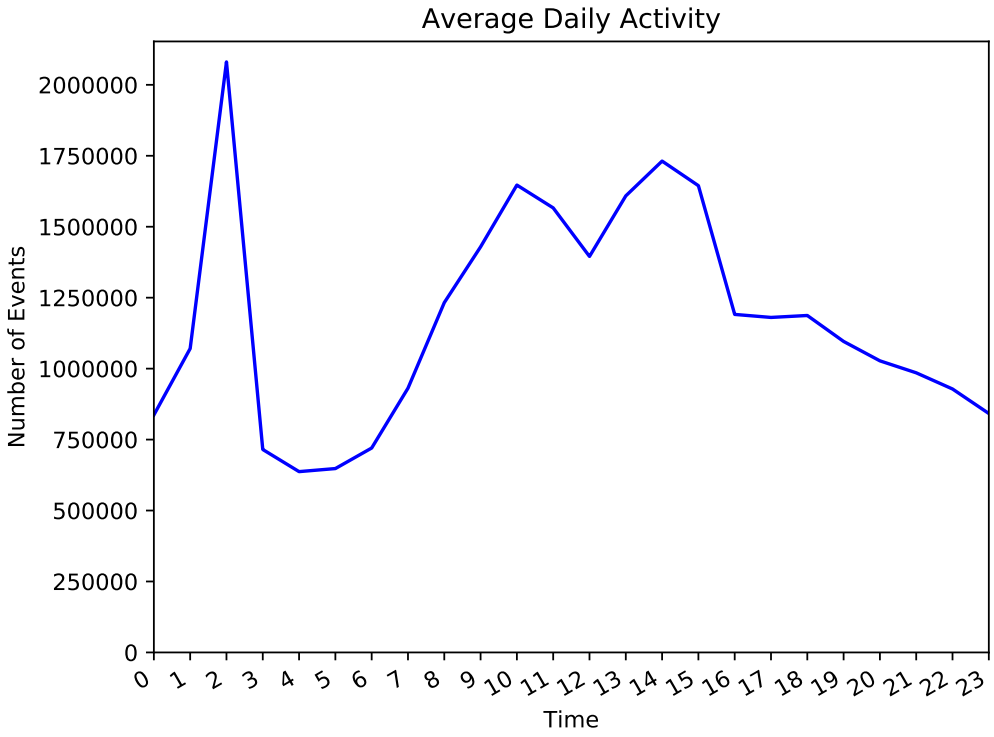

Figure 3.15: Average Daily Activity after Filter

11,296,413.

Figure 3.18 shows the PDF of user events over the week as a scatter plot. The y axis is the number of times a user performs an event. The x axis is the number of users that performed y number of events. Note that both axes are log scaled. This plot resembles an exponential decay in the number of events performed by users. The outliers are on the left end of the x axis. These individual users are the source of far more events than the rest of the users.

3.4.6 User Distribution across Domain Controllers

We next characterize user activity distributed across the campus domain controllers. We want to look at how many events are hitting each domain controller over time. Figure 3.19 shows the total number of events that target each domain controller over the week. Figure 3.20 illustrates the same but limited to unique users. Both graphs are binned into 24 hour periods.

The majority of all traffic we see is going through the UDCs each day. When considering only unique users, we see the same usage primarily through

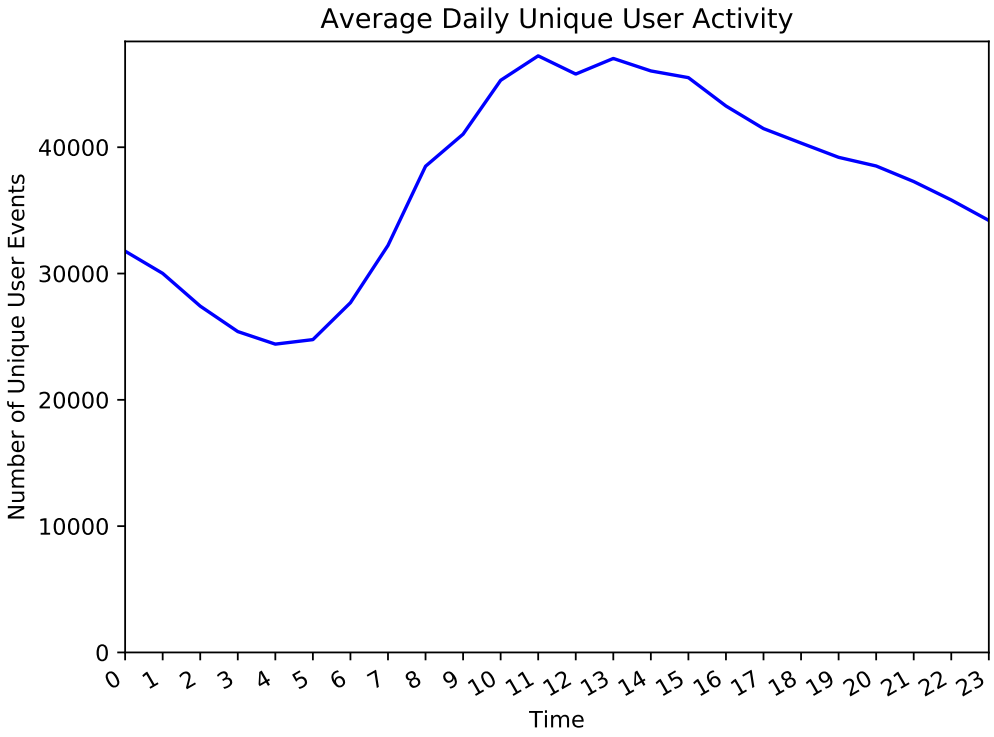

Figure 3.16: Average Unique Daily Activity after Filter

the UDCs. When we compare the total and the unique, we notice that total AWSDC and CDC traffic appears to be originating from fewer individuals.

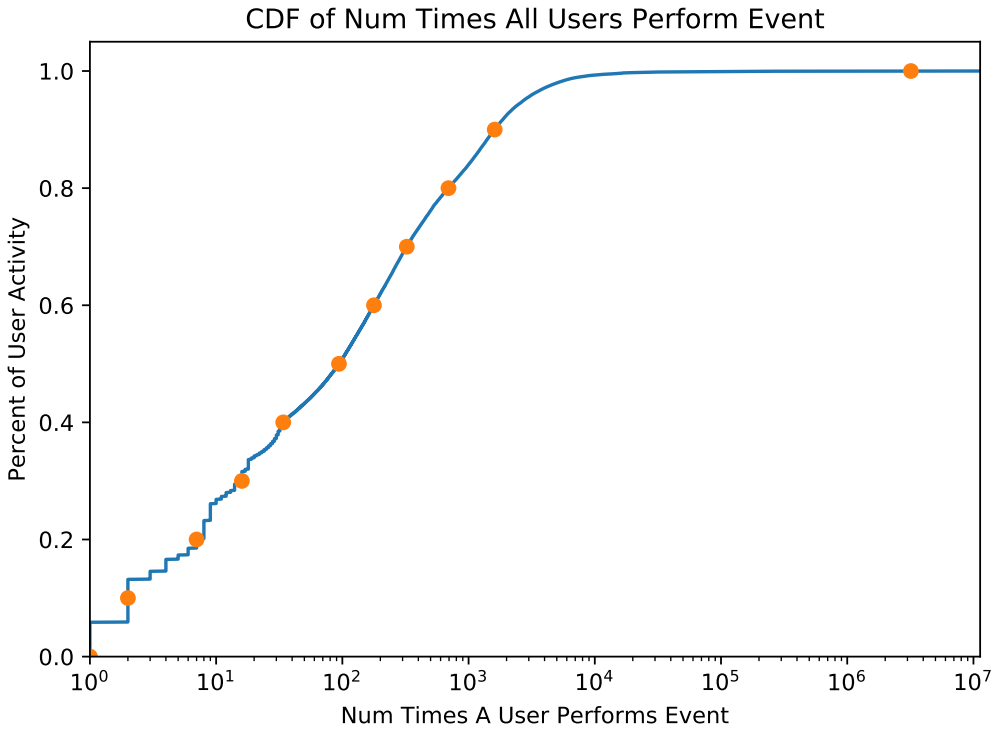

Figure 3.17: CDF of Number of Times All Users Login after Filter

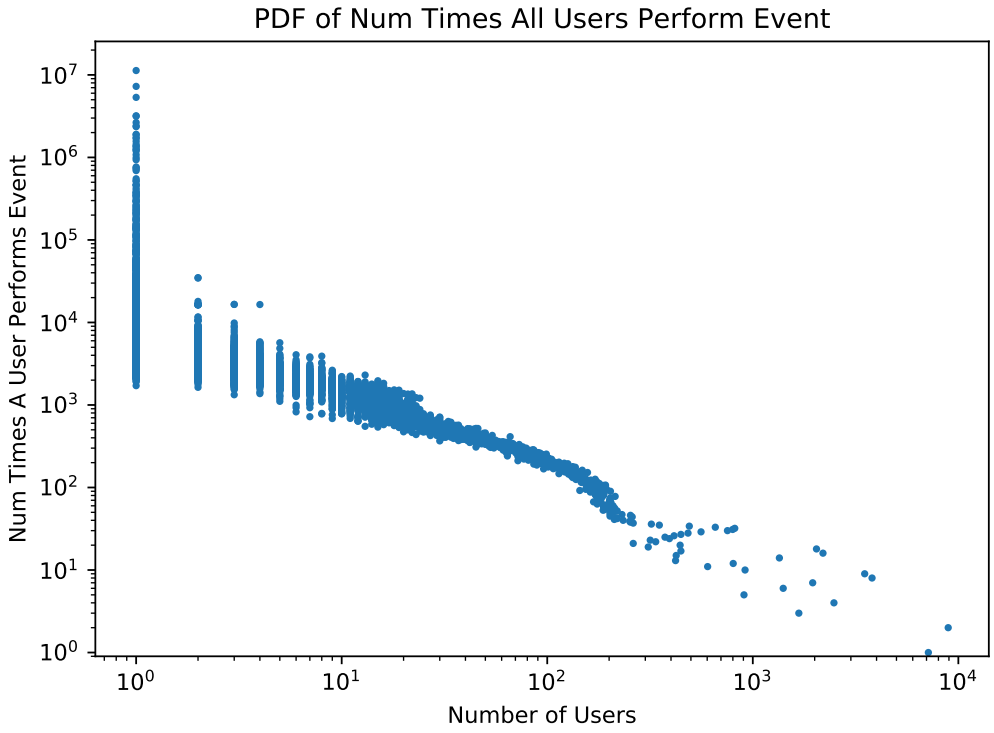

Figure 3.18: PDF of Number of Times All Users Login after Filter

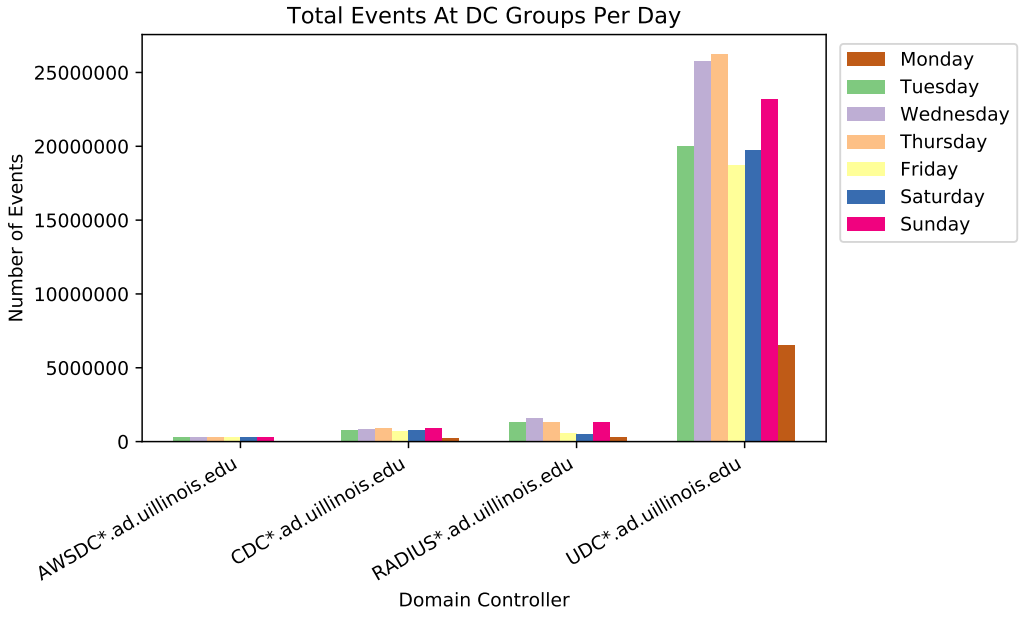

Figure 3.19: Total Number of Events at Each DC per Day after Filter

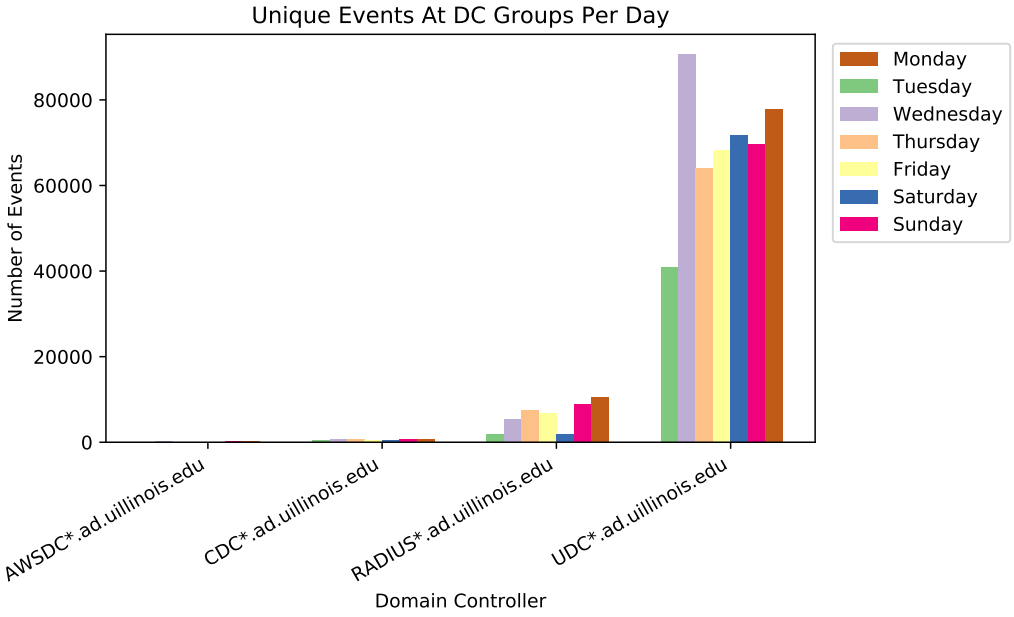

Figure 3.20: Number of Events from Unique Users at Each DC per Day after Filter

CHAPTER 4

CONCLUSION

4.1 Future Work

In the future, it would be useful to build models of how an attacker would move laterally in the network. We aim to gather data that an attacker would look for in order to gain access to other resources on a network. This includes, for example, Kerberos service tickets. We would assume that an attacker has already compromised a computer on the network and has access to a user account. The account may also have administrative privileges. The computer's memory would contain usernames, password hashes/keys, TGTs, and service tickets. An attacker with sufficient access can retrieve this information. The attacker can then use this information to authorize to other computers or resources, thus successfully moving laterally in a network.

With information about service tickets, we could build attack models of how an adversary could traverse a network. If we know how many service tickets are distributed to shared computers (i.e., a computer lab desktop), we can start to understand how many accounts could be impersonated through stolen tickets. The number of accounts on shared computers alone could also tell us how many account credentials could be stolen from hijacked hashes.

In addition, we want to look at user mobility. It would be beneficial to know how user movement appears on the network. This includes spatial-temporal movement. This would be useful for detecting lateral movement, for example, because mobility would put constraints on where an attacker could log in. A user should not normally be simultaneously logged in to and active at two computer labs that are across a campus. It would also be suspicious for a user to log out of one machine and immediately log in to another machine located on the other side of campus. These could both be indicators of an attacker attempting to use stolen credentials in order to

move laterally in the network.

4.2 Conclusion

In this thesis, we analyzed Windows event logs produced at Active Directory domain controllers. The network we examined was a university campus consisting of over 44,000 students and an additional 5,000 faculty and staff. The logs were over a week of network activity. We characterized the activity we saw from users, services, and shares. We described usage over a week, detailed usage based on event codes encountered, looked at daily average usage, and discussed distribution across the domain controllers. We then attempted to filter out the services and shares to focus on user activity alone. We repeated the previous analysis with the filtered data to gain additional insight on network behavior without services and shares. We saw that services and shares consist of a significant portion of network usage, including some of the outlier behavior.

Some of the data cannot be seen by looking at user activity alone. We saw that excluding services and shares also prevents us from seeing certain correlations, such as login and logout events. At the same time, services and shares can hinder us from seeing user activity by overshadowing users with spikes of activity.

The data supports multiple trends which are now reiterated. On average, the number of events created is lower in the early morning, increases towards noon, and starts to decline in the mid afternoon. Fewer events are generated on the weekend when compared to weekdays. In the filtered data, about 50% of users create 100 events or less and about 85% create 1,000 events or less. Less than 5% of users create more than 10,000 events. In both filtered and pre-filtered data, the top five events encountered are associated with user sessions (i.e., login, logout, authentication) or Kerberos ticket requests. Most events are generated at the Urbana Domain Controllers. The second largest number of events (although about 15 times smaller) are generated at the RADIUS DCs that serve only WiFi and VPN.

CHAPTER 5

REFERENCES

[1] K. Zetter, "An unprecedented look at Stuxnet, the world's first digital weapon," 2014, Wired. [Online]. Available: https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/

[2] M. Lennon, "Attackers increase use of PowerShell, WMI to evade detection: Mandiant," 2015, Security Week. [Online]. Available: http://www.securityweek.com/attackers-increase-use-powershell-wmi-evade-detection-mandiant

[3] W. Williamson, "Breaches are more than malware," 2015, Security Week. [Online]. Available: http://www.securityweek.com/breaches-are-more-malware

[4] Trend Micro, "Lateral movement: How do threat actors move deeper into your network?" 2013. [Online]. Available: http://about-threats.trendmicro.com/cloud-content/us/ent-primers/pdf/tlp_lateral_movement.pdf

[5] Trend Micro, "Understanding targeted attacks: Six components of targeted attacks," 2015. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/targeted-attacks-six-components

[6] Mitre, "Lateral movement," 2017. [Online]. Available: https://attack.mitre.org/wiki/Lateral_Movement

[7] R. Allen and A. Lowe-Norris, *Active Directory*. O'Reilly Media, Inc., 2003.

[8] Microsoft, "Kerberos explained," 2000. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb742516.aspx

[9] Microsoft, "How the kerberos version 5 authentication protocol works," 2009. [Online]. Available: https://technet.microsoft.com/en-us/library/cc772815(v=ws.10).aspx

[10] Microsoft, "Passwords technical overview," 2012. [Online]. Available: https://technet.microsoft.com/en-us/library/hh994558(v=ws.10).aspx

[11] J. Johansson, "Security watch the most misunderstood windows security setting of all time," 2006, Microsoft. [Online]. Available: https://technet.microsoft.com/en-us/library/2006.08.securitywatch.aspx

[12] Microsoft, "NTLM Overview," 2012. [Online]. Available: https://technet.microsoft.com/en-us/library/hh831571(v=ws.11).aspx

[13] Microsoft, "Changes in NTLM authentication," 2009. [Online]. Available: https://technet.microsoft.com/en-us/library/dd566199.aspx

[14] Microsoft, "Microsoft NTLM," 2000. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa378749(v=vs.85).aspx

[15] Cisco, "What is the difference between ntlm and ldap authentication?" 2014. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/security/web-security-appliance/118487-technote-wsa-00.html

[16] Microsoft, "What is kerberos authentication?" 2003. [Online]. Available: https://technet.microsoft.com/en-us/library/cc780469(v=ws.10).aspx

[17] Microsoft, "Maximum lifetime for user ticket," 2012. [Online]. Available: https://technet.microsoft.com/en-us/library/jj852169(v=ws.11).aspx

[18] Microsoft, "Maximum lifetime for service ticket," 2012. [Online]. Available: https://technet.microsoft.com/en-us/library/jj852188(v=ws.11).aspx

[19] C. Holmes, "Malware lateral movement: A primer," 2015, FireEye. [Online]. Available: https://www.fireeye.com/blog/executive-perspective/2015/08/malware_lateral_move.html

[20] J. Dunagan, A. X. Zheng, and D. R. Simon, "Heat-ray: combating identity snowball attacks using machinelearning, combinatorial optimization and attack graphs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 305–320.

[21] B. Delpy, "Github: mimikatz," 2014. [Online]. Available: https://github.com/gentilkiwi/mimikatz

[22] Microsoft, "Kerb_protocol_message_type enumeration," 2017. [Online]. Available: https://msdn.microsoft.com/library/windows/desktop/aa378099.aspx

[23] J. Lambert, "Defenders think in lists. attackers think in graphs. as long as this is true, attackers win." 2015. [Online]. Available: https://blogs.technet.microsoft.com/johnla/2015/04/26/defenders-think-in-lists-attackers-think-in-graphs-as-long-as-this-is-true-attackers-win/

[24] University of Illinois Public Affairs, "Campus facts," 2017. [Online]. Available: http://illinois.edu/about/facts.html

[25] D. Kotz and K. Essien, "Characterizing usage of a campus-wide wireless network," 2002.

[26] M. Balazinska and P. Castro, "Characterizing mobility and network usage in a corporate wireless local-area network," in *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, 2003, pp. 303–316.

[27] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," *Computer Networks*, vol. 52, no. 14, pp. 2690 – 2712, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128608001813

[28] I. Ullah, "Detecting Lateral Movement Attacks through SMB using BRO," M.S. thesis, University of Twente, 2016.

[29] M. M. Masud, L. Khan, B. Thuraisingham, X. Wang, P. Liu, and S. Zhu, "A data mining technique to detect remote exploits," in *Proc. IFIP WG 11.9 International Conference on Digital Forensics*, 2008, pp. 27–30.

[30] H. Siadati, B. Saket, and N. Memon, "Detecting malicious logins in enterprise networks using visualization," in *Visualization for Cyber Security (VizSec), 2016 IEEE Symposium on*. IEEE, 2016, pp. 1–8.

[31] J. R. Johnson and E. A. Hogan, "A graph analytic metric for mitigating advanced persistent threat," in *Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on*. IEEE, 2013, pp. 129–133.

[32] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[33] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan, "Characterizing user behavior and network performance in a public wireless lan," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1. ACM, 2002, pp. 195–205.

[34] F. PUB, "Secure hash standard (shs)," *FIPS PUB 180*, vol. 4, 2012.

[35] University of Illinois Technology Services, "U of I Box," 2017. [Online]. Available: https://techservices.illinois.edu/services/u-i-box/detail

[36] Box, "Github: box-python-sdk," 2017. [Online]. Available: https://github.com/box/box-python-sdk

[37] E. Jones, T. Oliphant, P. Peterson et al., "Numpy," 2017. [Online]. Available: http://www.numpy.org/

[38] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2017-11-06]. [Online]. Available: http://www.scipy.org/

[39] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[40] Monterey Technology Group, Inc., "Windows security log events," 2017. [Online]. Available: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx

APPENDIX A

LOG CONTENTS

We needed to know the exact contents of the logs we would be analyzing. We also needed to determine which fields in the log contained revealing information that must be anonymized. We received a sample log from Technology Services. This log contained information from only our NSRG lab volunteers. Table A.1 contains descriptions of each field we encountered in the sample log.

Table A.1: Field Descriptions

Name	Description
Account_Domain	The domain
Account_Name	Name of account just authenticated (when requesting TGT)
Additional_Information	Unknown
Authentication_Package	Always "MICROSOFT_AUTHENTICATION_PACKAGE_V1_0"
Client_Address	IP address of user
Client_Port	Source (user) port
ComputerName	Active Directory controller that received the request
Elevated_Token	Believed to be related to User Account Control (network admin account)
Error_Code	Integer code to describe the reason for an error
EventCode	Integer used to describe the event
EventType	Unknown (empty or 0)
Group_Domain	Domain of affected group
Group_Name	Name of affected group
Impersonation_Level	Unknown, added in Windows Server 2012

Table A.1 Continued

Name	Description
Key_Length	The length of the generated session key, will be 0 if no session key was requested.
Keywords	Seems to be Windows verification point, always "Audit Success"
Linked_Logon_ID	Unknown, believed to be linked to Transited Services
Logon_Account	Name of the account when NTLM authentication is used
Logon_GUID	Similar to Logon ID, but can potentially be correlated with event 4769 on Domain Controller
Logon_ID	Unique (between reboots) number for the logon session. Can be used to correlate backwards to logon event 4624
Logon_Process	Blank or "Advapi"
Logon_Type	How the user logged on, described below
NTLMErrCode	Unknown
Network_Account_Domain	Unknown (appears in 4624)
Network_Account_Name	Unknown (appears in 4624)
Network_Address	Same as Source_Network_Address
OpCode	Always "Info"
Package_Name__NTLM_only_	Which version of NTLM is used
Pre_Authentication_Type	Unknown
Privileges	Names of admin-equivalent privileges of user at logon
Process_ID	Executable process ID created from event 4688
Process_Name	Path of executable process created from event 4688
RecordNumber	Identifier for this transaction. Not unique across DCs
Restricted_Admin_Mode	"Yes" for Remote Desktop Connections where client specified this mode, "-" otherwise
Result_Code	An error code for TGT requests (details below)

Table A.1 Continued

Name	Description
Security_ID	SID of user account or affected group
Service_ID	Seems to always be empty
Service_Name	Always "krbtgt"
SourceName	Always "Microsoft Windows security auditing."
Source_Network_Address	IP address of user's computer but typically empty
Source_Port	Source TCP port of logon request (random)
Source_Workstation	Name of computer where logon attempt originated
Supplied_Realm_Name	Domain name of account
Target_Server_Name	Appears to be empty or "localhost"
TaskCategory	A brief string description of the event (not detailed)
Ticket_Encryption_Type	Unknown
Ticket_Options	Unknown
Transited_Services	Service acting on behalf of user for Kerberos authentication (client authenticates with service another way)
Type	Name of directory service, always "Information" for us
User_ID	SID of account used to login (TGT)
Virtual_Account	"Yes" when services are configured for this logon type, "No" otherwise
Workstation_Name	Computer name where user is physically present. In our case, requests are made on behalf of the user and so this becomes the DC
_time	Timestamp of when event was generated
host	Active Directory controller that received the request

The EventCode fields listed in Table A.2 are integer codes used to describe what event has occurred.

Table A.2: EventCode Descriptions

Event Code	Description
4624	An account was successfully logged on
4625	An account failed to log on
4634	An account was logged off [1]
4647	User initiated logoff
4648	A logon was attempted using explicit credentials
4672	Special privileges assigned to new logon
4673	A privileged service was called
4674	An operation was attempted on a privileged object
4675	SIDs were filtered
4716	Trusted domain information was modified
4720	A user account was created
4722	A user account was enabled
4723	An attempt was made to change an accountś password
4724	An attempt was made to reset an accounts password
4725	A user account was disabled
4727	A security-enabled global group was created
4728	A member was added to a security-enabled global group
4729	A member was removed from a security-enabled global group
4731	A security-enabled local group was created
4732	A member was added to a security-enabled local group
4734	A security-enabled local group was deleted
4735	A security-enabled local group was changed
4737	A security-enabled global group was changed

[1] "This event does not necessarily indicate the time that a user has stopped using a system. For example, if the computer is shut down or loses network connectivity it may not record a logoff event at all." [40]

Table A.2 Continued

Event Code	Description
4738	A user account was changed
4740	A user account was locked out
4755	A security-enabled universal group was changed
4756	A member was added to a security-enabled universal group
4757	A member was removed from a security-enabled universal group
4768	A Kerberos authentication ticket (TGT) was requested
4769	A Kerberos service ticket was requested
4771	Kerberos pre-authentication failed
4776	The domain controller attempted to validate the credentials for an account
4781	The name of an account was changed
4799	A security-enabled local group membership was enumerated[2]
4904	An attempt was made to register a security event source
4905	An attempt was made to unregister a security event source
4907	Auditing settings on object were changed
4946	A change has been made to Windows Firewall exception list. A rule was added
4948	A change has been made to Windows Firewall exception list. A rule was deleted
4954	Windows Firewall Group Policy settings has changed. The new settings have been applied
4985	The state of a transaction has changed
5038	Code integrity determined that the image hash of a file is not valid

[2] "This event is valuable for catching so-called APT actors who are scoping out the local accounts on a system they have compromised so that they extend their horizontal kill chain. Of course false positives are possible." [40]

Table A.2 Continued

Event Code	Description
5058	Key file operation
5059	Key migration operation
5136	A directory service object was modified
5137	A directory service object was created
5141	A directory service object was deleted

APPENDIX B

EVENT CODES ENCOUNTERED

B.1 Event Codes Before Filter

Table B.1 shows statistics about all event codes from all user activity over the one week data set. It contains the event code, total times encountered, individual users that have produced that event code, and the connections per user (total/unique).

Table B.1: EventCode Frequency

EventCode	Total Connections	Users	Connections Per User
4624	108078498	6569653	16.45117299
4634	108059073	6516457	16.58248846
4776	95899306	4640444	20.66597636
4768	46717178	6830640	6.839355902
4672	37356928	17929	2083.603547
4771	16948020	406600	41.68229218
4648	13203323	246895	53.47748233
4769	3891719	31941	121.8408628
5136	779206	4219	184.6897369
4625	494143	36980	13.36243916
4673	323799	1897	170.6900369
4733	274763	52660	5.217679453
4732	272926	53684	5.083935623
4735	153498	289	531.1349481
4756	141359	74908	1.887101511
4755	139976	342	409.2865497
4674	110029	630	174.6492063

Table B.1 Continued

EventCode	Total Connections	Users	Connections Per User
4675	57783	314	184.022293
4757	39153	20193	1.938939236
4740	37117	22597	1.642563172
4728	33561	15150	2.215247525
4737	30858	470	65.65531915
4738	5434	4801	1.131847532
4816	5244	277	18.93140794
4985	5128	646	7.938080495
4724	4954	4602	1.076488483
4723	1155	862	1.339907193
4729	1061	691	1.535455861
4722	1051	1026	1.024366472
4799	854	42	20.33333333
4907	633	1	633
4726	525	523	1.003824092
4781	439	49	8.959183673
4725	411	407	1.00982801
4720	380	376	1.010638298
4727	346	154	2.246753247
4904	198	65	3.046153846
4905	196	64	3.0625
4730	156	22	7.090909091
4754	113	38	2.973684211
4767	24	18	1.333333333
4731	23	11	2.090909091
4758	20	12	1.666666667
4954	14	2	7
4946	10	1	10
4948	10	1	10
5058	8	5	1.6
5061	8	5	1.6

Table B.1 Continued

EventCode	Total Connections	Users	Connections Per User
5137	5	1	5
4734	4	2	2
4764	4	2	2
4716	3	3	1
4647	2	2	1
5059	2	1	2
5038	1	1	1
5141	1	1	1

B.2 Event Codes Encountered After Filter

Table B.2 is similar to Table B.1, but applied to the data gathered after shares and services were filtered out.

Table B.2: EventCode Frequency After Filter

EventCode	Total Connections	Users	Connections Per User
4776	78763663	91315	862.5490117
4768	35507152	90913	390.5618778
4634	20675728	46444	445.1754371
4771	15566220	38394	405.4336615
4769	2748806	288	9544.465278
4733	270443	6507	41.56185646
4732	268594	6794	39.53400059
4756	140075	19313	7.252886657
4674	58864	1	58864
4757	38307	5422	7.065105127
4728	30506	11752	2.595813479
4737	29408	54	544.5925926
4672	22773	1	22773
4816	5289	1	5289

Table B.2 Continued

EventCode	Total Connections	Users	Connections Per User
4729	1067	471	2.265392781
4755	774	30	25.8
5136	699	21	33.28571429
4735	137	13	10.53846154
4738	65	49	1.326530612
4781	53	17	3.117647059
4723	35	27	1.296296296
4985	24	1	24
4724	23	19	1.210526316
4767	22	13	1.692307692
4954	14	1	14
4727	13	9	1.444444444
4731	12	2	6
4754	12	3	4
4946	10	1	10
4948	10	1	10
4720	5	5	1
4722	5	5	1
4730	4	3	1.333333333
4764	4	2	2
4716	3	1	3
5058	2	1	2
5059	2	1	2
5061	2	1	2
4725	1	1	1
4734	1	1	1
4758	1	1	1

APPENDIX C

CORRELATION COEFFICIENTS

C.1 Event Code Correlation Coefficients Before Filter

Table C.1 contains every combination of event code pairs and their correlation coefficients. The correlation coefficients were calculated using the Pandas (Python package) "coeff" function, which calculates the Pearson correlation coefficient between two variables. The calculation takes into account the time that the event occurred and amount of each event that occurred at that time.

Table C.1: EventCode Correlation Coefficients Before Filter

EventCode	EventCode	Correlation Coefficient
4648	4764	1
4674	4764	1
4675	4764	1
4722	4764	1
4724	4764	1
4725	4764	1
4726	4730	1
4726	4731	1
4726	4767	1
4727	4764	1
4728	4764	1
4730	4799	1
4730	4904	1
4730	4905	1
4731	4758	1
4738	4764	1

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4755	4764	1
4756	4764	1
4757	4764	1
4764	4767	1
5058	5061	1
4624	4634	0.999999295
4733	4735	0.999753925
4732	4735	0.99969289
4732	4733	0.99966972
4672	5058	0.999174778
4672	5061	0.999174778
4904	4905	0.998531774
4728	4737	0.997733244
4725	4730	0.984649292
4757	4781	0.98023896
4729	5058	0.977355555
4729	5061	0.977355555
4732	4781	0.964058999
4735	4781	0.960712481
4733	4781	0.955672795
4731	4737	0.95287689
4730	4733	0.944273473
4729	4731	0.911558399
4728	4731	0.907736977
4733	4757	0.903165522
4730	4735	0.900351388
4724	4738	0.896848452
4675	4769	0.893406711
4624	5058	0.873425875
4624	5061	0.873425875
4634	5058	0.872289324
4634	5061	0.872289324

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4634	4672	0.867899221
4624	4672	0.867838746
4673	4674	0.862043867
4756	4758	0.855349231
4767	4781	0.833561372
4720	4722	0.81564133
4768	4769	0.80322053
4727	5058	0.798007469
4727	5061	0.798007469
4724	4769	0.789314903
4754	4905	0.786795792
4675	4724	0.774424169
4724	4768	0.75575013
4728	5058	0.754605607
4728	5061	0.754605607
4672	4754	0.750945002
4675	4768	0.749302296
4732	5058	0.72097162
4732	5061	0.72097162
4723	4769	0.71994691
4733	4799	0.719340672
4722	4738	0.713105768
4758	4985	0.713069619
4730	5136	0.69954063
4634	4754	0.692171983
4624	4754	0.692144864
4754	4904	0.686406473
4799	4904	0.679980218
4675	4723	0.678999065
4768	4776	0.676386726
4738	4769	0.672349607
4733	4904	0.655246376

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4723	4724	0.649664721
4738	4768	0.648991192
4769	4776	0.628011373
4675	4738	0.625400361
4733	4905	0.595484633
4754	4756	0.594686783
4723	4768	0.59271681
4757	4758	0.591446592
4755	4767	0.586575459
4634	4776	0.585909402
4624	4776	0.585749061
4740	5058	0.581375697
4740	5061	0.581375697
4754	4776	0.57926166
4720	4727	0.568968537
4675	4776	0.564696563
4723	4738	0.562538792
4729	4781	0.556301257
4724	4776	0.543416956
4755	4799	0.54115842
4725	5136	0.537218737
4735	4757	0.536614566
4732	4757	0.536505529
4730	4738	0.533454088
4725	4767	0.528210755
4722	4724	0.526445545
4720	4738	0.525904226
4740	4768	0.518671711
4757	4799	0.518350289
4648	4776	0.511039604
4729	4733	0.50701591
4674	4731	0.501203959

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4756	4799	0.497610319
4728	4781	0.485025903
4731	4754	0.481534328
4720	4758	0.480277574
4722	4727	0.478834522
4738	4776	0.475490697
4735	5058	0.47256312
4735	5061	0.47256312
4725	4738	0.471162357
4727	4769	0.460486949
4722	4769	0.440729674
4723	4776	0.438702433
4625	4727	0.436991763
4731	4768	0.435095622
4754	4757	0.429822111
4740	4769	0.419377514
4767	4985	0.418819661
4735	4799	0.410340359
4624	4769	0.40973647
4634	4769	0.409674645
4675	4740	0.406075605
4729	4732	0.405319732
4729	4735	0.40517071
4732	4799	0.404024579
4675	4727	0.403435134
4737	5058	0.398261401
4737	5061	0.398261401
4799	4905	0.397445097
4740	4776	0.396758579
4624	4675	0.389291801
4634	4675	0.389106907
4725	4726	0.388179929

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4624	4648	0.383791594
4634	4648	0.38370429
4675	4722	0.379335885
4727	4738	0.379256617
4674	5058	0.376801806
4674	5061	0.376801806
4722	4776	0.375713221
4724	4727	0.368656954
4624	4724	0.367393899
4634	4724	0.367272619
4754	4758	0.364487074
4724	4740	0.357477293
4725	4754	0.355891182
4634	4768	0.354932117
4624	4768	0.354816612
4723	4740	0.354718819
4729	4757	0.354254542
4722	4768	0.350725256
4724	4767	0.348433297
4738	4799	0.346970443
4624	4723	0.341929435
4634	4723	0.341897491
4724	4799	0.335080573
4904	5136	0.330028585
4724	4755	0.328600677
4731	4816	0.327875848
4673	4768	0.319249948
4673	4731	0.318742479
4767	4769	0.316584658
4740	4767	0.312205474
4738	4740	0.307233399
4740	4755	0.306973527

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4634	4738	0.306222568
4624	4738	0.306208951
4738	4767	0.305111329
4905	5136	0.299951574
4735	4904	0.293451913
4732	4904	0.292057652
4724	4754	0.288783481
4727	4768	0.286862081
4767	4776	0.286382165
4723	4727	0.285731524
4768	4799	0.282162242
4624	4740	0.278406559
4634	4740	0.27837981
4674	4768	0.275688857
4755	4769	0.274421718
4675	4755	0.26907988
4723	4767	0.265381982
4673	4767	0.264352061
4738	4755	0.261893313
4735	4905	0.256283517
4732	4905	0.253920625
4727	4776	0.251187169
4625	4737	0.244540292
4731	4769	0.244267171
4726	5136	0.242550042
4648	5136	0.241975637
4720	4816	0.238146218
4723	4755	0.238072918
4625	4728	0.23786407
4727	4755	0.237732018
4722	4725	0.237283795
4673	4776	0.237053045

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4754	4769	0.2341658
4781	4816	0.231305226
4724	4756	0.230107588
4672	4776	0.229785152
4730	4776	0.223063777
4756	4757	0.221984112
4768	4771	0.221164861
4756	4776	0.220286093
4720	4724	0.213272803
4625	4816	0.212777779
4675	4799	0.212360341
4727	4767	0.212220478
4722	4723	0.211299967
4648	4725	0.21036714
4738	4754	0.208861289
4737	4755	0.208693259
4722	4731	0.205654475
4672	4740	0.205028335
4722	4754	0.204697611
4767	4768	0.199416053
4673	4740	0.198377985
4722	4767	0.192268516
4625	4673	0.191945164
4731	4756	0.191243307
4648	4730	0.19107515
4724	4771	0.190758237
4673	4675	0.190397653
4634	4722	0.189408151
4624	4722	0.1892055
4730	4754	0.188982237
4674	4740	0.187800095
4727	4740	0.185993386

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4720	4985	0.181867817
4776	4799	0.18129938
4729	4816	0.180240694
4755	4768	0.179396774
4725	4768	0.177426732
4672	4724	0.176883799
4674	4720	0.176592871
4673	4769	0.175386399
4731	4781	0.175058708
4733	4758	0.174077656
4725	4769	0.173011802
4731	4776	0.172531344
4816	4904	0.170840873
4672	4723	0.169858732
4725	4776	0.168992868
4731	4755	0.165841101
4731	5136	0.16459944
4720	4725	0.15872896
4720	4730	0.158270148
4674	4816	0.156012623
4672	4758	0.154177794
4672	4738	0.153825103
4625	4769	0.151922702
4625	4771	0.151713642
4672	4720	0.151447325
4672	4769	0.149725366
4675	4754	0.149462529
4722	4816	0.14879263
4738	4771	0.147060943
4675	4756	0.146579764
4722	4755	0.144469037
4754	4768	0.144252225

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4674	4776	0.141982225
4740	4799	0.14156647
4625	4731	0.139587468
4816	4905	0.138442573
4738	4756	0.138107399
4648	4767	0.137373739
4730	4768	0.137349338
4674	4675	0.13082937
4727	4771	0.130236186
4673	4730	0.127999458
4673	4720	0.127371488
4673	4758	0.126827749
4672	4675	0.125534797
4673	4725	0.125163313
4731	4757	0.120109142
4724	4725	0.119715155
4673	4755	0.116697946
4730	4816	0.114997345
4672	4722	0.114942782
4722	4740	0.114758278
4733	4756	0.113823225
4625	4720	0.113091119
4674	4755	0.113030891
4723	4725	0.112737494
4722	5136	0.110557041
4781	4985	0.108611071
4720	4776	0.105572013
4625	4674	0.103838077
4731	4985	0.100900241
4756	4769	0.10037523
4672	4768	0.099080219
4799	5136	0.098884008

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4723	4771	0.097416324
4672	4756	0.09268911
4625	4723	0.091132999
4624	4673	0.08766845
4634	4673	0.087601441
4624	4755	0.083938755
4634	4755	0.083642591
4673	4816	0.083199959
4624	4756	0.082302635
4634	4756	0.082181872
4634	4727	0.081487402
4624	4727	0.081392926
4756	4768	0.080045945
4634	5136	0.078095811
4725	4740	0.077922105
4624	5136	0.077823121
4634	4720	0.077260336
4673	4726	0.077194845
4624	4720	0.077111109
4756	4767	0.075926603
4725	4816	0.075752967
4625	4675	0.075561402
4723	4799	0.074263701
4625	4722	0.073948409
4755	4776	0.072081607
4673	4722	0.071080917
4728	4904	0.067931087
4730	4732	0.067604335
4769	4799	0.067590037
4674	4781	0.067268834
4634	4799	0.066902289
4624	4799	0.066901414

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4735	4738	0.065046292
4675	5058	0.064397183
4675	5061	0.064397183
4737	4754	0.062876567
4724	4731	0.062743811
4730	4769	0.062598086
4625	4725	0.061975495
4625	4756	0.06110192
4675	4767	0.059650349
4771	4776	0.057709309
4724	4735	0.057089223
4720	4756	0.054369343
4732	4756	0.053894919
4735	4756	0.053774948
4673	4723	0.052873393
4648	4675	0.050776664
4732	4738	0.050618806
4674	4758	0.049944538
4673	4985	0.049144269
4730	4740	0.0480964
4776	5136	0.04758154
4625	4768	0.046751234
4674	4722	0.046491714
4732	4816	0.046472971
4673	4727	0.044830493
4724	4732	0.043571386
4755	4756	0.042499096
4675	4904	0.042207525
4720	4754	0.041887711
4725	4731	0.041364612
4725	4727	0.040272249
4674	4769	0.039768651

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4735	4816	0.038115917
4728	4905	0.037575802
4816	4985	0.037408681
4673	5136	0.036308163
4723	4816	0.035344024
4769	5058	0.035187061
4769	5061	0.035187061
4648	4673	0.034622704
4720	5136	0.032875416
4737	4816	0.0328664
4672	4905	0.032700653
4672	4904	0.032576405
4625	4738	0.031895708
4648	4769	0.029771686
4771	4799	0.029768887
4723	4729	0.028111981
4674	4767	0.027858724
4673	4732	0.027497784
4673	4735	0.026839222
4731	4738	0.026457596
4674	4732	0.026301875
4674	4735	0.026049042
4674	4985	0.025659864
4728	4816	0.025329008
4729	4740	0.025216066
4731	4740	0.02510296
4720	4740	0.024904441
4675	4735	0.024845105
4723	4756	0.024775524
4738	5136	0.024674366
4672	4755	0.024371931
4757	4767	0.024330061

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4625	4755	0.024322767
4674	4726	0.022100279
4729	4738	0.021959385
4672	4726	0.020893051
4769	5136	0.018011152
4674	4756	0.017950986
4728	4754	0.017893927
4624	4904	0.01781188
4634	4904	0.017769323
4740	4754	0.017745528
4754	4767	0.016643567
4672	4729	0.015416136
4674	4727	0.013990234
4733	4816	0.013861604
4648	4731	0.013461438
4675	4732	0.01307817
4673	4728	0.012699798
4985	5058	0.011252739
4985	5061	0.011252739
4738	4816	0.011103938
4625	4724	0.010713641
4737	4904	0.009243371
4634	4725	0.008822104
4624	4725	0.008497016
4634	4771	0.007938435
4624	4771	0.007632877
4675	4905	0.00754022
4675	4725	0.007202784
4725	4756	0.00691714
4673	4737	0.006272822
4648	4768	0.006020458
4634	4758	0.005880357

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4624	4758	0.005769958
4767	4816	0.005040993
4723	5136	0.004871086
4720	4755	0.004574977
4740	4756	0.003833943
4728	4733	0.003078946
4648	4674	0.002996722
4724	4729	0.00253969
4722	4730	0.002048217
4722	4756	0.002038896
4755	4985	0.001887543
4673	4754	0.001600228
4727	5136	0.001540657
4672	4799	0.000541979
4624	4647	0
4624	4716	0
4624	4734	0
4624	4907	0
4624	4946	0
4624	4948	0
4624	4954	0
4624	5038	0
4624	5059	0
4624	5137	0
4624	5141	0
4625	4647	0
4625	4716	0
4625	4734	0
4625	4907	0
4625	4946	0
4625	4948	0
4625	4954	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4625	5038	0
4625	5059	0
4625	5137	0
4625	5141	0
4634	4647	0
4634	4716	0
4634	4734	0
4634	4907	0
4634	4946	0
4634	4948	0
4634	4954	0
4634	5038	0
4634	5059	0
4634	5137	0
4634	5141	0
4647	4648	0
4647	4672	0
4647	4673	0
4647	4674	0
4647	4675	0
4647	4716	0
4647	4720	0
4647	4722	0
4647	4723	0
4647	4724	0
4647	4725	0
4647	4726	0
4647	4727	0
4647	4728	0
4647	4729	0
4647	4730	0
4647	4731	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4647	4732	0
4647	4733	0
4647	4734	0
4647	4735	0
4647	4737	0
4647	4738	0
4647	4740	0
4647	4754	0
4647	4755	0
4647	4756	0
4647	4757	0
4647	4758	0
4647	4764	0
4647	4767	0
4647	4768	0
4647	4769	0
4647	4771	0
4647	4776	0
4647	4781	0
4647	4799	0
4647	4816	0
4647	4904	0
4647	4905	0
4647	4907	0
4647	4946	0
4647	4948	0
4647	4954	0
4647	4985	0
4647	5038	0
4647	5058	0
4647	5059	0
4647	5061	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4647	5136	0
4647	5137	0
4647	5141	0
4648	4716	0
4648	4734	0
4648	4907	0
4648	4946	0
4648	4948	0
4648	4954	0
4648	5038	0
4648	5059	0
4648	5137	0
4648	5141	0
4672	4716	0
4672	4734	0
4672	4907	0
4672	4946	0
4672	4948	0
4672	4954	0
4672	5038	0
4672	5059	0
4672	5137	0
4672	5141	0
4673	4716	0
4673	4734	0
4673	4907	0
4673	4946	0
4673	4948	0
4673	4954	0
4673	5038	0
4673	5059	0
4673	5137	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4673	5141	0
4674	4716	0
4674	4734	0
4674	4907	0
4674	4946	0
4674	4948	0
4674	4954	0
4674	5038	0
4674	5059	0
4674	5137	0
4674	5141	0
4675	4716	0
4675	4734	0
4675	4907	0
4675	4946	0
4675	4948	0
4675	4954	0
4675	5038	0
4675	5059	0
4675	5137	0
4675	5141	0
4716	4720	0
4716	4722	0
4716	4723	0
4716	4724	0
4716	4725	0
4716	4726	0
4716	4727	0
4716	4728	0
4716	4729	0
4716	4730	0
4716	4731	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4716	4732	0
4716	4733	0
4716	4734	0
4716	4735	0
4716	4737	0
4716	4738	0
4716	4740	0
4716	4754	0
4716	4755	0
4716	4756	0
4716	4757	0
4716	4758	0
4716	4764	0
4716	4767	0
4716	4768	0
4716	4769	0
4716	4771	0
4716	4776	0
4716	4781	0
4716	4799	0
4716	4816	0
4716	4904	0
4716	4905	0
4716	4907	0
4716	4946	0
4716	4948	0
4716	4954	0
4716	4985	0
4716	5038	0
4716	5058	0
4716	5059	0
4716	5061	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4716	5136	0
4716	5137	0
4716	5141	0
4720	4734	0
4720	4764	0
4720	4907	0
4720	4946	0
4720	4948	0
4720	4954	0
4720	5038	0
4720	5059	0
4720	5137	0
4720	5141	0
4722	4734	0
4722	4907	0
4722	4946	0
4722	4948	0
4722	4954	0
4722	5038	0
4722	5059	0
4722	5137	0
4722	5141	0
4723	4734	0
4723	4907	0
4723	4946	0
4723	4948	0
4723	4954	0
4723	5038	0
4723	5059	0
4723	5137	0
4723	5141	0
4724	4734	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4724	4907	0
4724	4946	0
4724	4948	0
4724	4954	0
4724	5038	0
4724	5059	0
4724	5137	0
4724	5141	0
4725	4734	0
4725	4907	0
4725	4946	0
4725	4948	0
4725	4954	0
4725	5038	0
4725	5059	0
4725	5137	0
4725	5141	0
4726	4727	0
4726	4734	0
4726	4754	0
4726	4758	0
4726	4764	0
4726	4904	0
4726	4905	0
4726	4907	0
4726	4946	0
4726	4948	0
4726	4954	0
4726	5038	0
4726	5058	0
4726	5059	0
4726	5061	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4726	5137	0
4726	5141	0
4727	4734	0
4727	4907	0
4727	4946	0
4727	4948	0
4727	4954	0
4727	5038	0
4727	5059	0
4727	5137	0
4727	5141	0
4728	4734	0
4728	4907	0
4728	4946	0
4728	4948	0
4728	4954	0
4728	5038	0
4728	5059	0
4728	5137	0
4728	5141	0
4729	4734	0
4729	4907	0
4729	4946	0
4729	4948	0
4729	4954	0
4729	5038	0
4729	5059	0
4729	5137	0
4729	5141	0
4730	4731	0
4730	4734	0
4730	4758	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4730	4764	0
4730	4767	0
4730	4907	0
4730	4946	0
4730	4948	0
4730	4954	0
4730	5038	0
4730	5058	0
4730	5059	0
4730	5061	0
4730	5137	0
4730	5141	0
4731	4734	0
4731	4764	0
4731	4907	0
4731	4946	0
4731	4948	0
4731	4954	0
4731	5038	0
4731	5058	0
4731	5059	0
4731	5061	0
4731	5137	0
4731	5141	0
4732	4734	0
4732	4764	0
4732	4907	0
4732	4946	0
4732	4948	0
4732	4954	0
4732	5038	0
4732	5059	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4732	5137	0
4732	5141	0
4733	4734	0
4733	4764	0
4733	4907	0
4733	4946	0
4733	4948	0
4733	4954	0
4733	5038	0
4733	5059	0
4733	5137	0
4733	5141	0
4734	4735	0
4734	4737	0
4734	4738	0
4734	4740	0
4734	4754	0
4734	4755	0
4734	4756	0
4734	4757	0
4734	4758	0
4734	4764	0
4734	4767	0
4734	4768	0
4734	4769	0
4734	4771	0
4734	4776	0
4734	4781	0
4734	4799	0
4734	4816	0
4734	4904	0
4734	4905	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4734	4907	0
4734	4946	0
4734	4948	0
4734	4954	0
4734	4985	0
4734	5038	0
4734	5058	0
4734	5059	0
4734	5061	0
4734	5136	0
4734	5137	0
4734	5141	0
4735	4764	0
4735	4907	0
4735	4946	0
4735	4948	0
4735	4954	0
4735	5038	0
4735	5059	0
4735	5137	0
4735	5141	0
4737	4907	0
4737	4946	0
4737	4948	0
4737	4954	0
4737	5038	0
4737	5059	0
4737	5137	0
4737	5141	0
4738	4907	0
4738	4946	0
4738	4948	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4738	4954	0
4738	5038	0
4738	5059	0
4738	5137	0
4738	5141	0
4740	4907	0
4740	4946	0
4740	4948	0
4740	4954	0
4740	5038	0
4740	5059	0
4740	5137	0
4740	5141	0
4754	4907	0
4754	4946	0
4754	4948	0
4754	4954	0
4754	5038	0
4754	5059	0
4754	5137	0
4754	5141	0
4755	4907	0
4755	4946	0
4755	4948	0
4755	4954	0
4755	5038	0
4755	5059	0
4755	5137	0
4755	5141	0
4756	4907	0
4756	4946	0
4756	4948	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4756	4954	0
4756	5038	0
4756	5059	0
4756	5137	0
4756	5141	0
4757	4907	0
4757	4946	0
4757	4948	0
4757	4954	0
4757	5038	0
4757	5059	0
4757	5137	0
4757	5141	0
4758	4764	0
4758	4799	0
4758	4904	0
4758	4905	0
4758	4907	0
4758	4946	0
4758	4948	0
4758	4954	0
4758	5038	0
4758	5058	0
4758	5059	0
4758	5061	0
4758	5137	0
4758	5141	0
4764	4781	0
4764	4799	0
4764	4904	0
4764	4905	0
4764	4907	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4764	4946	0
4764	4948	0
4764	4954	0
4764	5038	0
4764	5058	0
4764	5059	0
4764	5061	0
4764	5137	0
4764	5141	0
4767	4799	0
4767	4907	0
4767	4946	0
4767	4948	0
4767	4954	0
4767	5038	0
4767	5058	0
4767	5059	0
4767	5061	0
4767	5137	0
4767	5141	0
4768	4907	0
4768	4946	0
4768	4948	0
4768	4954	0
4768	5038	0
4768	5059	0
4768	5137	0
4768	5141	0
4769	4907	0
4769	4946	0
4769	4948	0
4769	4954	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4769	5038	0
4769	5059	0
4769	5137	0
4769	5141	0
4771	4907	0
4771	4946	0
4771	4948	0
4771	4954	0
4771	5038	0
4771	5059	0
4771	5137	0
4771	5141	0
4776	4907	0
4776	4946	0
4776	4948	0
4776	4954	0
4776	5038	0
4776	5059	0
4776	5137	0
4776	5141	0
4781	4907	0
4781	4946	0
4781	4948	0
4781	4954	0
4781	5038	0
4781	5058	0
4781	5059	0
4781	5061	0
4781	5137	0
4781	5141	0
4799	4907	0
4799	4946	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4799	4948	0
4799	4954	0
4799	5038	0
4799	5059	0
4799	5137	0
4799	5141	0
4816	4907	0
4816	4946	0
4816	4948	0
4816	4954	0
4816	5038	0
4816	5059	0
4816	5137	0
4816	5141	0
4904	4907	0
4904	4946	0
4904	4948	0
4904	4954	0
4904	5038	0
4904	5058	0
4904	5059	0
4904	5061	0
4904	5137	0
4904	5141	0
4905	4907	0
4905	4946	0
4905	4948	0
4905	4954	0
4905	5038	0
4905	5058	0
4905	5059	0
4905	5061	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4905	5137	0
4905	5141	0
4907	4946	0
4907	4948	0
4907	4954	0
4907	4985	0
4907	5038	0
4907	5058	0
4907	5059	0
4907	5061	0
4907	5136	0
4907	5137	0
4907	5141	0
4946	4948	0
4946	4954	0
4946	4985	0
4946	5038	0
4946	5058	0
4946	5059	0
4946	5061	0
4946	5136	0
4946	5137	0
4946	5141	0
4948	4954	0
4948	4985	0
4948	5038	0
4948	5058	0
4948	5059	0
4948	5061	0
4948	5136	0
4948	5137	0
4948	5141	0

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4954	4985	0
4954	5038	0
4954	5058	0
4954	5059	0
4954	5061	0
4954	5136	0
4954	5137	0
4954	5141	0
4985	5038	0
4985	5059	0
4985	5137	0
4985	5141	0
5038	5058	0
5038	5059	0
5038	5061	0
5038	5136	0
5038	5137	0
5038	5141	0
5058	5059	0
5058	5137	0
5058	5141	0
5059	5061	0
5059	5136	0
5059	5137	0
5059	5141	0
5061	5137	0
5061	5141	0
5136	5137	0
5136	5141	0
5137	5141	0
4648	4754	-0.001079675
4625	4776	-0.001276717

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4756	4781	-0.002508843
4816	5136	-0.002898104
4648	4727	-0.002971993
4673	4781	-0.002998589
4675	4731	-0.003842919
4624	4905	-0.00406142
4634	4905	-0.004269516
4729	4755	-0.004651017
4674	4730	-0.005026857
4754	5136	-0.005145724
4735	4755	-0.00569882
4674	4725	-0.005934305
4625	4985	-0.006076056
4732	4755	-0.006658528
4673	4756	-0.006705878
4674	4729	-0.008736865
4624	4729	-0.009170715
4720	4728	-0.009382701
4634	4729	-0.009500368
4672	4985	-0.00958047
4728	4730	-0.009596867
4674	5136	-0.01011211
4769	4904	-0.010153682
4674	4728	-0.010239376
4720	4737	-0.011033375
4769	4771	-0.012245137
4648	4723	-0.013580179
4674	4737	-0.01377802
4648	4672	-0.014103033
4737	4756	-0.015225214
4674	4757	-0.018621615
4737	4985	-0.019228705

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4728	4985	-0.019510888
4732	4985	-0.019543479
4673	4729	-0.019913256
4672	5136	-0.020830055
4722	4771	-0.02097266
4985	5136	-0.021404302
4625	4740	-0.022028865
4648	4771	-0.022935374
4723	4754	-0.023165931
4737	4905	-0.023175332
4675	5136	-0.024153555
4799	4985	-0.024635654
4727	4816	-0.024888788
4624	4674	-0.025370416
4634	4674	-0.025379245
4720	4768	-0.025437189
4648	4816	-0.025811036
4729	5136	-0.026731827
4672	4816	-0.027000173
4735	4985	-0.027075954
4674	4799	-0.029367045
4727	4754	-0.029476706
4672	4673	-0.030037493
4722	4904	-0.030913562
4737	4771	-0.031007217
4673	4738	-0.031339631
4723	4735	-0.032194751
4672	4757	-0.032550349
4771	4985	-0.032701214
4672	4771	-0.032951326
4675	4729	-0.033620728
4740	4816	-0.035070872

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4723	4730	-0.035945489
4648	4724	-0.036595785
4723	4732	-0.037541511
4672	4727	-0.03788286
4625	4634	-0.038610706
4624	4625	-0.038668347
4672	4725	-0.039045612
4728	4771	-0.039167757
4675	4985	-0.039268854
4720	4905	-0.039528089
4728	4767	-0.039859306
4769	4905	-0.042611686
4771	4816	-0.043092402
4675	4771	-0.043417937
4757	4985	-0.043858051
4756	4985	-0.04469237
4754	4755	-0.044746332
4625	4672	-0.045769286
4769	4985	-0.046782507
4727	4904	-0.047576188
4720	4904	-0.04846831
4625	5136	-0.048906038
4672	4737	-0.049186414
4648	4738	-0.049324616
4624	4985	-0.050269836
4634	4985	-0.050453
4648	4729	-0.050771616
4729	4768	-0.051197216
4755	5136	-0.05285945
4724	4730	-0.053408141
4672	4728	-0.053703006
4722	4905	-0.053891295

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4729	4985	-0.054236055
4648	4722	-0.054389302
4673	4724	-0.054862519
4768	4985	-0.055239219
4735	4769	-0.055911958
4648	4985	-0.056116271
4754	4781	-0.056130507
4648	4740	-0.056663887
4625	4754	-0.05731357
4722	4985	-0.059105107
4648	4755	-0.060866496
4727	4905	-0.06116558
4776	4816	-0.061542041
4755	4816	-0.061757792
4735	4740	-0.061961354
4674	4723	-0.062355903
4648	4756	-0.062828356
4673	4757	-0.063571144
4722	5058	-0.063757671
4722	5061	-0.063757671
4740	4985	-0.064649405
4724	4985	-0.064748373
4728	4732	-0.065508418
4728	4735	-0.066074283
4732	4737	-0.066955623
4735	4737	-0.067512313
4634	4816	-0.067660885
4624	4816	-0.06769041
4732	4769	-0.067731884
4723	4985	-0.068207495
4771	4904	-0.068484872
4672	4735	-0.068826815

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4672	4732	-0.069415634
4738	4985	-0.069631186
4727	4985	-0.071122425
4757	4771	-0.072075255
4625	4767	-0.072362434
4625	4730	-0.073374347
4732	4740	-0.074502096
4727	4737	-0.075899665
4768	5136	-0.076229142
4725	4905	-0.076768155
4729	4756	-0.078373605
4737	4767	-0.079235122
4725	4771	-0.081139904
4624	4735	-0.081526676
4634	4735	-0.08214159
4771	4905	-0.082261618
4625	4904	-0.08320254
4720	4771	-0.083204943
4727	4728	-0.083718168
4725	4904	-0.084693484
4729	4754	-0.084817523
4720	4767	-0.08483883
4674	4771	-0.087208535
4720	4769	-0.08786717
4722	4737	-0.088615379
4624	4732	-0.088982749
4634	4732	-0.08959418
4648	4720	-0.091731275
4724	4733	-0.091982682
4675	4737	-0.09242516
4724	5136	-0.09242837
4725	4728	-0.095055714

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4733	4737	-0.095159636
4725	4758	-0.095870624
4725	4985	-0.096463422
4725	4737	-0.096654617
4672	4733	-0.097295468
4728	4755	-0.097418911
4722	4728	-0.097820731
4733	4738	-0.097949337
4725	4755	-0.098757191
4733	4985	-0.098933888
4727	4756	-0.0991509
4757	4816	-0.09954508
4756	4904	-0.100542203
4799	4816	-0.100637342
4674	4738	-0.101225891
4733	4755	-0.101585099
4675	4728	-0.104775319
4724	4904	-0.105120926
4735	4768	-0.106901057
4738	4904	-0.10725049
4625	4905	-0.107813204
4648	4735	-0.107832126
4724	4816	-0.108127488
4673	4771	-0.10825955
4732	4767	-0.109612498
4735	4767	-0.110104769
4729	4769	-0.110547319
4737	4740	-0.111066035
4648	4799	-0.111967808
4726	4985	-0.112616686
4776	4904	-0.113671168
4771	5136	-0.114035149

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4675	4720	-0.115009432
4776	4985	-0.115072841
4648	4732	-0.11535845
4674	4724	-0.116254084
4672	4674	-0.117019554
4758	5136	-0.11733403
4768	4816	-0.118712486
4625	4648	-0.118835677
4735	4771	-0.119902092
4737	4769	-0.120872992
4732	4771	-0.122462417
4740	5136	-0.123697738
4728	4740	-0.12398341
4723	4731	-0.124360877
4737	4757	-0.125515506
4723	4737	-0.126898834
4732	4768	-0.127673995
4754	4985	-0.127766818
4769	4816	-0.128775416
4674	4733	-0.128846932
4728	5136	-0.13001122
4755	4757	-0.133066497
4725	4729	-0.133101684
4723	4728	-0.133134734
4675	4816	-0.133282438
4727	4735	-0.133916106
4728	4769	-0.133918967
4624	4757	-0.13469999
4755	4771	-0.134788339
4634	4757	-0.135110969
4730	4985	-0.135130362
4727	4732	-0.135905884

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4673	4905	-0.136586155
4648	4737	-0.136969155
4725	4735	-0.137460476
4737	4799	-0.141588126
4733	4767	-0.143612206
4729	4737	-0.143624065
4624	4737	-0.143976671
4634	4737	-0.144169048
4634	4730	-0.145475586
4625	4732	-0.146017446
4648	4728	-0.146073978
4624	4730	-0.146130946
4738	4905	-0.146144046
4722	4735	-0.146882178
4722	4732	-0.14695087
4724	4905	-0.146959134
4726	4732	-0.147245788
4776	4905	-0.147295332
4754	4771	-0.14790629
4625	4735	-0.148484928
4726	4735	-0.14874405
4729	4758	-0.14887392
4728	4756	-0.149871148
4737	5136	-0.151573571
4756	4905	-0.151616959
4720	4731	-0.153109466
4756	4771	-0.153276405
4728	4729	-0.154765763
4624	4728	-0.155159131
4756	4816	-0.155328613
4634	4728	-0.155355663
4673	4904	-0.1556067

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4755	4781	-0.15769804
4725	4757	-0.158080112
4729	4904	-0.159960737
4729	4776	-0.161118207
4725	4732	-0.163123149
4732	5136	-0.163531194
4735	5136	-0.164433583
4767	5136	-0.166070461
4757	4904	-0.166759727
4624	4733	-0.167272479
4634	4733	-0.168078157
4737	4738	-0.168348764
4728	4758	-0.168763246
4675	4730	-0.172506201
4729	4905	-0.172608072
4724	4737	-0.173626512
4723	4904	-0.174172971
4727	4731	-0.174503157
4726	4781	-0.175118199
4673	5058	-0.17621671
4673	5061	-0.17621671
4674	4905	-0.178361842
4730	4771	-0.179031539
4730	4755	-0.180159004
4737	4781	-0.180738162
4728	4757	-0.181663089
4728	4738	-0.184417367
4722	4799	-0.185937781
4675	4733	-0.186005763
4673	4799	-0.187269785
4738	4757	-0.189876179
4726	4756	-0.190392162

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4724	4728	-0.190529973
4725	4733	-0.192509447
4733	4771	-0.193874593
4905	4985	-0.196961339
4674	4904	-0.197355535
4768	4904	-0.198158129
4724	4757	-0.198362867
4731	4771	-0.19914427
4732	4754	-0.199402597
4648	4904	-0.200218341
4740	4904	-0.200742986
4735	4754	-0.201628068
4625	4729	-0.201665897
4720	4735	-0.202293387
4720	4723	-0.202719733
4720	4732	-0.203929601
4722	4729	-0.205092802
4768	5058	-0.205752259
4768	5061	-0.205752259
4740	4771	-0.209701103
4672	4730	-0.210915322
4672	4731	-0.211939308
4740	4905	-0.212136231
4674	4754	-0.216043161
4729	4771	-0.221195273
4726	4755	-0.223445408
4723	4905	-0.22344899
4767	4771	-0.22367134
4624	4767	-0.226088448
4648	4905	-0.226270445
4634	4767	-0.22628406
4723	4733	-0.227962841

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4672	4781	-0.229551322
4735	4776	-0.230335699
4733	4754	-0.231929355
4720	4729	-0.23212749
4729	4730	-0.233887914
4720	4757	-0.235737054
4730	4737	-0.236127934
4726	4771	-0.237980436
4758	4776	-0.238991179
4737	4776	-0.240251467
4768	4905	-0.240666025
4756	5136	-0.240847127
4754	4816	-0.242802596
4625	4799	-0.24348849
4730	4757	-0.244579689
4723	4757	-0.252213804
4732	4776	-0.253060111
4737	4768	-0.253597518
4728	4776	-0.257878048
4722	4758	-0.260010834
4673	4733	-0.261549567
4729	4799	-0.261979198
4648	4757	-0.262187264
4733	4740	-0.262419545
4727	4729	-0.26321804
4730	4756	-0.263765521
4757	4905	-0.266175081
4771	4781	-0.267967213
4672	4767	-0.269674777
4731	4732	-0.270294052
4726	4816	-0.270606486
4728	4768	-0.27350865

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4634	4731	-0.273583757
4624	4731	-0.273678684
4675	4757	-0.275188965
4729	4767	-0.279654188
4634	4726	-0.280786574
4624	4726	-0.280998463
4755	4758	-0.281321251
4731	4735	-0.281576021
4904	4985	-0.282140296
4737	4758	-0.284457622
4740	4757	-0.289659845
4720	4799	-0.292504217
4625	4757	-0.29973468
4720	4781	-0.305703991
4733	4769	-0.313286797
4757	4776	-0.321306882
5058	5136	-0.321876114
5061	5136	-0.321876114
4757	4769	-0.322218708
4754	4799	-0.325179072
4731	4767	-0.333333333
4731	4904	-0.333333333
4725	4781	-0.333373213
4727	4757	-0.337458926
4720	4733	-0.337675568
4733	4768	-0.342877813
4726	4733	-0.343182364
4771	5058	-0.345591511
4771	5061	-0.345591511
4757	4768	-0.348627031
4727	4799	-0.349561009
4722	4757	-0.350776449

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4648	4733	-0.352902965
4625	5058	-0.352946969
4625	5061	-0.352946969
4723	5058	-0.354329336
4723	5061	-0.354329336
4755	4904	-0.370735428
4720	4726	-0.375113584
4731	4733	-0.376627088
4727	4781	-0.377900541
4733	5136	-0.392312742
4727	4730	-0.392647635
4624	4781	-0.393220366
4634	4781	-0.393766168
4767	4904	-0.395032854
4767	4905	-0.395032854
4727	4733	-0.404055049
4728	4799	-0.412310451
4740	4781	-0.41500989
4758	4769	-0.417008036
4733	4776	-0.423239748
4755	4905	-0.425426166
4722	4733	-0.4296183
4757	5136	-0.432076724
4733	5058	-0.435455109
4733	5061	-0.435455109
4758	4781	-0.442108287
4740	4758	-0.445363527
4726	4729	-0.447713524
4758	4771	-0.447751512
4738	4758	-0.449774144
4730	4781	-0.457495711
4731	4905	-0.457495711

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4738	4781	-0.464285542
4755	5058	-0.469118224
4755	5061	-0.469118224
4756	5058	-0.470835177
4756	5061	-0.470835177
4757	5058	-0.473400496
4757	5061	-0.473400496
4776	5058	-0.481750636
4776	5061	-0.481750636
4722	4781	-0.486308497
4723	4781	-0.488680691
4724	4758	-0.499884308
4720	5058	-0.5
4720	5061	-0.5
4724	5058	-0.517333977
4724	5061	-0.517333977
4735	4758	-0.518477386
4625	4726	-0.518711301
4726	4740	-0.522342124
4738	5058	-0.546987308
4738	5061	-0.546987308
4625	4733	-0.555024869
4781	5136	-0.56218363
4726	4757	-0.570375421
4758	4767	-0.577350269
4758	4816	-0.58468133
4816	5058	-0.587890754
4816	5061	-0.587890754
4724	4781	-0.591467581
4723	4726	-0.598536416
4732	4758	-0.600200697
4726	4737	-0.60876241

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4725	4799	-0.614660239
4722	4726	-0.619215197
4781	4904	-0.628312119
4781	4905	-0.628312119
4723	4758	-0.668089274
4726	4768	-0.67090959
4675	4758	-0.676866654
4648	5058	-0.682481192
4648	5061	-0.682481192
4625	4758	-0.69913745
4758	4768	-0.699160698
4726	4776	-0.705328872
4648	4726	-0.713356799
4648	4758	-0.714950325
4731	4799	-0.71759845
4726	4769	-0.728217556
4726	4728	-0.7330605
4724	4726	-0.736273119
4727	4758	-0.745367689
4769	4781	-0.74662156
4726	4738	-0.754093571
4781	4799	-0.760330514
4648	4781	-0.775919689
4625	4781	-0.783686157
4776	4781	-0.789688702
4675	4781	-0.857207855
4725	5058	-0.866025404
4725	5061	-0.866025404
4768	4781	-0.872724927
4675	4726	-0.894780258
4624	4764	-1
4625	4764	-1

Table C.1 Continued

EventCode	EventCode	Correlation Coefficient
4634	4764	-1
4672	4764	-1
4673	4764	-1
4723	4764	-1
4726	4799	-1
4729	4764	-1
4737	4764	-1
4740	4764	-1
4754	4764	-1
4754	5058	-1
4754	5061	-1
4764	4768	-1
4764	4769	-1
4764	4771	-1
4764	4776	-1
4764	4816	-1
4764	4985	-1
4764	5136	-1
4799	5058	-1
4799	5061	-1

C.2 Event Code Correlation Coefficients After Filter

Table C.2 is similar to Table C.1, but applied to the data gathered after shares and services were filtered out.

Table C.2: EventCode Correlation Coefficients After Filter

EventCode	EventCode	Correlation Coefficient
4634	4764	1
4672	4764	1
4674	4716	1
4674	4764	1

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4716	4729	1
4716	4733	1
4716	4735	1
4716	4737	1
4720	4722	1
4720	4735	1
4722	4735	1
4728	4764	1
4731	4738	1
4738	4754	1
4754	4767	1
4755	4764	1
4756	4764	1
4757	4764	1
4764	4767	1
4764	5136	1
4728	4737	0.99991538
4730	4733	0.999814866
4732	4733	0.999800279
4754	4816	0.999792452
4672	4720	0.999669585
4672	4722	0.999669585
4672	4716	0.999368509
4720	4737	0.997256511
4722	4737	0.997256511
4674	4754	0.996114031
4754	5136	0.996078416
4728	4754	0.994659663
4716	4728	0.994367475
4754	4771	0.993307386
4754	4756	0.988632273
4674	4720	0.984498777

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4674	4722	0.984498777
4716	4771	0.980751481
4672	4754	0.980742322
4720	4757	0.976307064
4722	4757	0.976307064
4672	4674	0.973282133
4720	4768	0.970151768
4722	4768	0.970151768
4720	5136	0.965294349
4722	5136	0.965294349
4720	4771	0.962246548
4722	4771	0.962246548
4737	4755	0.959654727
4720	4756	0.952713359
4722	4756	0.952713359
4716	4768	0.952065246
4634	4716	0.945979288
4634	4720	0.94545827
4634	4722	0.94545827
4716	4769	0.926441061
4733	4757	0.924744734
4754	4768	0.924565977
4727	4735	0.924500327
4634	4754	0.923990667
4720	4816	0.912720384
4722	4816	0.912720384
4716	4816	0.908128264
4730	4771	0.904374043
4723	4755	0.898354065
4728	4730	0.850439435
4754	4776	0.822753484
4768	4769	0.798124368

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4716	4776	0.798010118
4724	4738	0.794007962
4720	4776	0.772546201
4722	4776	0.772546201
4727	4781	0.755928946
4754	4769	0.754505825
4768	4776	0.748865583
4674	4768	0.746549547
4723	4738	0.725167484
4720	4769	0.723599049
4722	4769	0.723599049
4634	4776	0.719301731
4672	4768	0.716969552
4729	4755	0.716709457
4732	4754	0.715320239
4735	4755	0.675114776
4769	4776	0.673518014
4724	4735	0.673189067
4716	4732	0.671931944
4768	4771	0.665210343
4732	4757	0.65706034
4728	4755	0.637378891
4720	4781	0.628618557
4722	4781	0.628618557
4731	5136	0.625
4672	4724	0.615454059
4729	4733	0.614683899
4767	4781	0.610541276
4674	4724	0.603298232
4672	4771	0.602309793
4674	4771	0.593954906
4756	4781	0.588974951

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4720	4724	0.577350269
4722	4724	0.577350269
4728	4733	0.572567834
4771	5136	0.571036481
4727	4733	0.568903075
4720	4729	0.567613479
4722	4729	0.567613479
4674	4738	0.559464108
4727	5136	0.541130832
4674	4816	0.525925683
4729	4732	0.517616176
4757	4781	0.517600674
4672	4738	0.513587189
4738	4755	0.501628099
4672	4816	0.497594154
4729	4757	0.486512253
4738	5136	0.465387145
4771	4776	0.462366355
4776	5136	0.458087917
4768	5136	0.457995984
4733	4738	0.457832441
4634	4768	0.456813639
4738	4781	0.452858923
4674	4776	0.434463771
4724	4729	0.420897867
4672	4776	0.419657944
4634	4672	0.410811148
4634	4674	0.403010956
4634	4769	0.394809084
4634	5136	0.393272992
4738	4767	0.390199486
4769	5136	0.388873893

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4720	4728	0.388378667
4722	4728	0.388378667
4727	4732	0.385462897
4769	4771	0.373789748
4732	4738	0.371963127
4674	4769	0.364572774
4723	4735	0.358568583
4729	4816	0.358050833
4738	4757	0.35207576
4735	4738	0.349506319
4737	4781	0.34661018
4733	4816	0.340040676
4672	4769	0.339885473
4738	4768	0.333966889
4724	4767	0.333333333
4724	4757	0.332542985
4724	4737	0.329345691
4674	4729	0.328162648
4634	4771	0.325764459
4672	5136	0.324121768
4781	5136	0.322056595
4674	5136	0.310658974
4723	5136	0.305041064
4756	5136	0.302383182
4729	4731	0.29743804
4672	4729	0.295015934
4767	5136	0.281966908
4723	4737	0.281204282
4767	4769	0.277355625
4727	4737	0.252318084
4728	4757	0.249608869
4727	4755	0.247770372

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4756	4757	0.24180747
4737	4738	0.237941291
4768	4816	0.237445011
4732	4816	0.236901115
4767	4776	0.220502773
4729	4768	0.219766677
4729	4735	0.218984606
4672	4735	0.214240145
4738	4771	0.210097103
4738	4756	0.207596079
4674	4733	0.199665595
4672	4733	0.199532652
4724	4781	0.193649167
4727	4816	0.189652562
4674	4732	0.188772925
4724	4771	0.188735231
4733	4756	0.18686247
4735	4781	0.18267282
4756	4776	0.179872172
4674	4735	0.176777412
4634	4767	0.172298989
4672	4732	0.164290693
4727	4728	0.161271082
4723	4781	0.158113883
4674	4757	0.157107393
4723	4771	0.155290766
4735	4816	0.154788252
4738	4769	0.15232293
4634	4738	0.149478033
4757	4816	0.149185502
4771	4816	0.148191705
4728	4738	0.135515014

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4724	4816	0.13442604
4738	4776	0.133709564
4674	4723	0.133031205
4720	4738	0.132453236
4722	4738	0.132453236
4776	4781	0.129479927
4634	4729	0.128003552
4735	4768	0.121393096
4756	4768	0.119033855
4634	4816	0.11677128
4724	4756	0.115051684
4724	5136	0.110611529
4735	4756	0.110606134
4756	4769	0.107114859
4723	4768	0.099870837
4728	4781	0.092701539
4674	4756	0.090742175
4767	4768	0.09061182
4729	4781	0.088321724
4634	4756	0.086226127
4672	4757	0.083431263
4735	4771	0.080668316
4776	4816	0.078658579
4728	4767	0.078607856
4735	4776	0.078224206
4672	4756	0.078071457
4732	4756	0.072829785
4674	4767	0.072604419
4723	4728	0.067644287
4723	4729	0.06718098
4767	4816	0.064718378
4757	4767	0.0564968

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4634	4781	0.055851229
4729	4776	0.052640868
4728	4756	0.051928361
4674	4781	0.051328091
4755	4816	0.051313125
4732	4768	0.04981891
4723	4776	0.049800478
4634	4723	0.049519775
4755	4757	0.045740656
4737	4816	0.045102809
4634	4735	0.044246572
4735	4757	0.043525367
4729	4771	0.041754643
4769	4781	0.040272534
4737	4756	0.037956556
4731	4735	0.035245369
4728	5136	0.034569289
4724	4768	0.032374504
4768	4781	0.030911461
4767	4771	0.029018724
4733	4755	0.028085501
4732	4781	0.027344141
4737	4757	0.022820466
4729	4738	0.020465575
4735	5136	0.019588524
4672	4781	0.018050587
4672	4727	0.012946885
4769	4816	0.011527089
4756	4816	0.011469622
4756	4767	0.007763285
4634	4725	0
4634	4734	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4634	4758	0
4634	4946	0
4634	4948	0
4634	4954	0
4634	4985	0
4634	5058	0
4634	5059	0
4634	5061	0
4672	4725	0
4672	4734	0
4672	4758	0
4672	4946	0
4672	4948	0
4672	4954	0
4672	4985	0
4672	5058	0
4672	5059	0
4672	5061	0
4674	4725	0
4674	4730	0
4674	4734	0
4674	4758	0
4674	4946	0
4674	4948	0
4674	4954	0
4674	4985	0
4674	5058	0
4674	5059	0
4674	5061	0
4716	4720	0
4716	4722	0
4716	4723	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4716	4724	0
4716	4725	0
4716	4727	0
4716	4730	0
4716	4731	0
4716	4734	0
4716	4738	0
4716	4754	0
4716	4758	0
4716	4764	0
4716	4767	0
4716	4781	0
4716	4946	0
4716	4948	0
4716	4954	0
4716	4985	0
4716	5058	0
4716	5059	0
4716	5061	0
4720	4723	0
4720	4725	0
4720	4727	0
4720	4730	0
4720	4731	0
4720	4733	0
4720	4734	0
4720	4754	0
4720	4758	0
4720	4764	0
4720	4946	0
4720	4948	0
4720	4954	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4720	4985	0
4720	5058	0
4720	5059	0
4720	5061	0
4722	4723	0
4722	4725	0
4722	4727	0
4722	4730	0
4722	4731	0
4722	4733	0
4722	4734	0
4722	4754	0
4722	4758	0
4722	4764	0
4722	4946	0
4722	4948	0
4722	4954	0
4722	4985	0
4722	5058	0
4722	5059	0
4722	5061	0
4723	4725	0
4723	4727	0
4723	4730	0
4723	4731	0
4723	4734	0
4723	4754	0
4723	4758	0
4723	4764	0
4723	4767	0
4723	4946	0
4723	4948	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4723	4954	0
4723	4985	0
4723	5058	0
4723	5059	0
4723	5061	0
4724	4725	0
4724	4727	0
4724	4728	0
4724	4730	0
4724	4731	0
4724	4734	0
4724	4754	0
4724	4758	0
4724	4764	0
4724	4946	0
4724	4948	0
4724	4954	0
4724	4985	0
4724	5058	0
4724	5059	0
4724	5061	0
4725	4727	0
4725	4728	0
4725	4729	0
4725	4730	0
4725	4731	0
4725	4732	0
4725	4733	0
4725	4734	0
4725	4735	0
4725	4737	0
4725	4738	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4725	4754	0
4725	4755	0
4725	4756	0
4725	4757	0
4725	4758	0
4725	4764	0
4725	4767	0
4725	4768	0
4725	4769	0
4725	4771	0
4725	4776	0
4725	4781	0
4725	4816	0
4725	4946	0
4725	4948	0
4725	4954	0
4725	4985	0
4725	5058	0
4725	5059	0
4725	5061	0
4725	5136	0
4727	4730	0
4727	4731	0
4727	4734	0
4727	4738	0
4727	4754	0
4727	4758	0
4727	4764	0
4727	4767	0
4727	4946	0
4727	4948	0
4727	4954	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4727	4985	0
4727	5058	0
4727	5059	0
4727	5061	0
4728	4734	0
4728	4758	0
4728	4946	0
4728	4948	0
4728	4954	0
4728	4985	0
4728	5058	0
4728	5059	0
4728	5061	0
4729	4734	0
4729	4758	0
4729	4946	0
4729	4948	0
4729	4954	0
4729	4985	0
4729	5058	0
4729	5059	0
4729	5061	0
4730	4731	0
4730	4734	0
4730	4735	0
4730	4738	0
4730	4754	0
4730	4758	0
4730	4764	0
4730	4767	0
4730	4781	0
4730	4946	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4730	4948	0
4730	4954	0
4730	4985	0
4730	5058	0
4730	5059	0
4730	5061	0
4731	4734	0
4731	4754	0
4731	4755	0
4731	4758	0
4731	4764	0
4731	4767	0
4731	4946	0
4731	4948	0
4731	4954	0
4731	4985	0
4731	5058	0
4731	5059	0
4731	5061	0
4732	4734	0
4732	4758	0
4732	4764	0
4732	4946	0
4732	4948	0
4732	4954	0
4732	4985	0
4732	5058	0
4732	5059	0
4732	5061	0
4733	4734	0
4733	4758	0
4733	4764	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4733	4946	0
4733	4948	0
4733	4954	0
4733	4985	0
4733	5058	0
4733	5059	0
4733	5061	0
4734	4735	0
4734	4737	0
4734	4738	0
4734	4754	0
4734	4755	0
4734	4756	0
4734	4757	0
4734	4758	0
4734	4764	0
4734	4767	0
4734	4768	0
4734	4769	0
4734	4771	0
4734	4776	0
4734	4781	0
4734	4816	0
4734	4946	0
4734	4948	0
4734	4954	0
4734	4985	0
4734	5058	0
4734	5059	0
4734	5061	0
4734	5136	0
4735	4758	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4735	4764	0
4735	4946	0
4735	4948	0
4735	4954	0
4735	4985	0
4735	5058	0
4735	5059	0
4735	5061	0
4737	4754	0
4737	4758	0
4737	4764	0
4737	4946	0
4737	4948	0
4737	4954	0
4737	4985	0
4737	5058	0
4737	5059	0
4737	5061	0
4738	4758	0
4738	4764	0
4738	4946	0
4738	4948	0
4738	4954	0
4738	4985	0
4738	5058	0
4738	5059	0
4738	5061	0
4754	4758	0
4754	4764	0
4754	4946	0
4754	4948	0
4754	4954	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4754	4985	0
4754	5058	0
4754	5059	0
4754	5061	0
4755	4758	0
4755	4946	0
4755	4948	0
4755	4954	0
4755	4985	0
4755	5058	0
4755	5059	0
4755	5061	0
4756	4758	0
4756	4946	0
4756	4948	0
4756	4954	0
4756	4985	0
4756	5058	0
4756	5059	0
4756	5061	0
4757	4758	0
4757	4946	0
4757	4948	0
4757	4954	0
4757	4985	0
4757	5058	0
4757	5059	0
4757	5061	0
4758	4764	0
4758	4767	0
4758	4768	0
4758	4769	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4758	4771	0
4758	4776	0
4758	4781	0
4758	4816	0
4758	4946	0
4758	4948	0
4758	4954	0
4758	4985	0
4758	5058	0
4758	5059	0
4758	5061	0
4758	5136	0
4764	4781	0
4764	4946	0
4764	4948	0
4764	4954	0
4764	4985	0
4764	5058	0
4764	5059	0
4764	5061	0
4767	4946	0
4767	4948	0
4767	4954	0
4767	4985	0
4767	5058	0
4767	5059	0
4767	5061	0
4768	4946	0
4768	4948	0
4768	4954	0
4768	4985	0
4768	5058	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4768	5059	0
4768	5061	0
4769	4946	0
4769	4948	0
4769	4954	0
4769	4985	0
4769	5058	0
4769	5059	0
4769	5061	0
4771	4946	0
4771	4948	0
4771	4954	0
4771	4985	0
4771	5058	0
4771	5059	0
4771	5061	0
4776	4946	0
4776	4948	0
4776	4954	0
4776	4985	0
4776	5058	0
4776	5059	0
4776	5061	0
4781	4946	0
4781	4948	0
4781	4954	0
4781	4985	0
4781	5058	0
4781	5059	0
4781	5061	0
4816	4946	0
4816	4948	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4816	4954	0
4816	4985	0
4816	5058	0
4816	5059	0
4816	5061	0
4946	4948	0
4946	4954	0
4946	4985	0
4946	5058	0
4946	5059	0
4946	5061	0
4946	5136	0
4948	4954	0
4948	4985	0
4948	5058	0
4948	5059	0
4948	5061	0
4948	5136	0
4954	4985	0
4954	5058	0
4954	5059	0
4954	5061	0
4954	5136	0
4985	5058	0
4985	5059	0
4985	5061	0
4985	5136	0
5058	5059	0
5058	5061	0
5058	5136	0
5059	5061	0
5059	5136	0

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
5061	5136	0
4672	4767	-6.92E-05
4729	4769	-0.00051772
4732	4735	-0.002926929
4728	4816	-0.003172485
4672	4723	-0.004261153
4723	4756	-0.004742711
4732	4771	-0.007342178
4729	4756	-0.008285018
4737	4767	-0.010414711
4733	4771	-0.013285454
4674	4727	-0.015792713
4738	4816	-0.019620252
4672	4728	-0.023930936
4727	4771	-0.02433118
4733	4768	-0.024537612
4737	5136	-0.024591159
4731	4771	-0.026804283
4728	4731	-0.030923453
4731	4816	-0.031598603
4720	4732	-0.031958098
4722	4732	-0.031958098
4816	5136	-0.032066736
4727	4768	-0.036128737
4733	4735	-0.037268493
4781	4816	-0.040775056
4732	4769	-0.040795051
4723	4769	-0.046222826
4735	4767	-0.048029211
4732	4755	-0.049344031
4757	4771	-0.053686582
4728	4732	-0.056911085

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4728	4771	-0.062511485
4756	4771	-0.06281842
4755	4781	-0.06559696
4755	4771	-0.067789129
4674	4728	-0.072608634
4723	4757	-0.07823312
4634	4732	-0.079349059
4672	4755	-0.080133212
4755	4756	-0.081617862
4674	4755	-0.084774968
4672	4737	-0.086653166
4757	4768	-0.091415285
4732	4737	-0.105056116
4732	5136	-0.107682728
4634	4755	-0.10948934
4724	4732	-0.113750144
4755	5136	-0.113763651
4737	4771	-0.125762699
4755	4768	-0.129147544
4727	4776	-0.129907821
4757	5136	-0.132304664
4728	4769	-0.132320094
4729	5136	-0.132539037
4731	4768	-0.133806035
4723	4732	-0.135449778
4771	4781	-0.140056487
4723	4733	-0.142423665
4728	4735	-0.142448373
4674	4737	-0.143066275
4732	4776	-0.15186712
4634	4733	-0.153997495
4672	4730	-0.162757692

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4727	4756	-0.164123749
4731	4737	-0.174108878
4728	4729	-0.175823119
4729	4737	-0.17641269
4735	4769	-0.177149643
4634	4728	-0.178278343
4728	4768	-0.181461759
4634	4757	-0.191203382
4674	4731	-0.196132354
4724	4733	-0.199217417
4634	4727	-0.209656875
4733	4769	-0.213068018
4723	4816	-0.213991481
4732	4767	-0.221028637
4735	4737	-0.224620443
4757	4776	-0.225548801
4733	4781	-0.227988481
4634	4731	-0.229519362
4733	4737	-0.234186196
4728	4776	-0.236024055
4727	4729	-0.242350939
4733	4776	-0.245656276
4733	5136	-0.250903914
4757	4769	-0.252887124
4755	4776	-0.253698007
4672	4731	-0.257975591
4731	4776	-0.266370813
4755	4769	-0.271725828
4729	4767	-0.273680212
4733	4767	-0.283493895
4755	4767	-0.303583612
4727	4769	-0.311962751

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4731	4732	-0.314696137
4724	4755	-0.353392543
4727	4757	-0.353953174
4720	4755	-0.381246426
4722	4755	-0.381246426
4730	4816	-0.38765486
4737	4768	-0.39325503
4730	4737	-0.397359707
4731	4756	-0.402846499
4737	4769	-0.40310898
4731	4733	-0.404860098
4634	4724	-0.422541184
4731	4757	-0.42291027
4634	4737	-0.428515337
4724	4776	-0.431873876
4754	4755	-0.445424897
4735	4754	-0.459781196
4730	5136	-0.461083968
4737	4776	-0.482320689
4723	4724	-0.5
4724	4769	-0.532677263
4731	4781	-0.555555556
4730	4755	-0.654653671
4730	4757	-0.654653671
4730	4768	-0.7059679
4731	4769	-0.753083696
4730	4776	-0.761900692
4729	4730	-0.917662935
4730	4732	-0.933256525
4730	4756	-0.933256525
4634	4730	-0.944726543
4730	4769	-0.99986289

Table C.2 Continued

EventCode	EventCode	Correlation Coefficient
4716	4755	-1
4716	4756	-1
4716	4757	-1
4716	5136	-1
4720	4767	-1
4722	4767	-1
4729	4754	-1
4729	4764	-1
4733	4754	-1
4754	4757	-1
4754	4781	-1
4764	4768	-1
4764	4769	-1
4764	4771	-1
4764	4776	-1
4764	4816	-1