AN ANALYSIS OF KERBEROASTING ATTACK AND DETECTION WITH SUPERVISED MACHINE LEARNING ALGORITHMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

YASİN AKSÜT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF CYBER SECURITY

NOVEMBER 2024

An Analysis Of Kerberoasting Attack And Detection With Supervised Machine Learning Algorithms

submitted by **YASİN AKSÜT** in partial fulfillment of the requirements for the degree of **Master of Science in Cyber Security Department, Middle East Technical University** by,

Prof. Dr. Banu GÜNEL KILIÇ _____
Dean, **Graduate School of Informatics**

Assoc. Prof. Dr. Cihangir TEZCAN _____
Head of Department, **Cyber Security**

Assoc. Prof. Dr. Cihangir TEZCAN _____
Supervisor, **Graduate School of Informatics, METU**

Examining Committee Members:

Prof. Dr. Ali Aydın Selçuk
Computer Engineering, TOBB University _____

Assoc. Prof. Dr. Cihangir TEZCAN
Graduate School of Informatics, METU _____

Assist. Prof. Dr. Aybar Can Acar
Graduate School of Informatics, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: YASİN AKSÜT

Signature :

ABSTRACT

AN ANALYSIS OF KERBEROASTING ATTACK AND DETECTION WITH SUPERVISED MACHINE LEARNING ALGORITHMS

AKSÜT, YASİN

M.S., Department of Cyber Security

Supervisor: Assoc. Prof. Dr. Cihangir TEZCAN

November 2024, 81 pages

Active Directory (AD) is one of the most widely used directory services today, playing a key role in organizing and managing network resources within an organization. In cybersecurity, AD serves as a significant component for defense in depth, offering layered security by controlling access to network assets, enforcing authentication policies, and monitoring for suspicious activity. Therefore, it is essential to have a robust security strategy in place to prevent and detect AD attacks in depth. Detection of AD attacks is difficult because attackers often use techniques that blend in with normal network traffic and activities. Among the AD attacks, Kerberoasting attack which leverages inherent weaknesses in the Kerberos authentication protocol used by AD can be most stealthy and may not exhibit obvious signs of compromise. It makes it difficult for security teams to detect them using traditional security tools. In this work, we are going to try to provide a solution for detection of Kerberoasting attacks by using supervised machine learning algorithms. Moreover, there is no publicly available dataset that can be used to measure the efficiency of any machine learning algorithm for Kerberoasting attacks for the sake of protecting the security of sensitive data. For this reason, we created a dataset by conducting the study in a virtual environment and we made security logs publicly available.

Keywords: Active Directory Security, Kerberos Protocol, Supervised Machine Learning Algorithms

ÖZ

KERBEROASTİNG SALDIRISININ ANALİZİ VE DENETİMLİ MAKİNA ÖĞRENMESİ ALGORİTMALARI İLE TESPİTİ

AKSÜT, YASİN

Yüksek Lisans, Siber Güvenlik Bölümü

Tez Yöneticisi: Doç. Dr. Cihangir TEZCAN

Kasım 2024, 81 sayfa

Active Directory (AD), günümüzde en yaygın kullanılan dizin hizmetlerinden biridir ve bir organizasyon içindeki ağ kaynaklarını organize etme ve yönetmede önemli bir rol oynamaktadır. Siber güvenlikte AD, ağ varlıklarına erişimi kontrol eder, kimlik doğrulama politikalarını uygular ve şüpheli etkinlikleri izleyerek katmanlı güvenlik sunmak suretiyle derinlemesine savunma için önemli bir bileşen görevi görür. Bu nedenle, AD saldırılarını derinlemesine önlemek ve tespit etmek için sağlam bir güvenlik stratejisine sahip olmak önemlidir. AD saldırılarının tespiti zordur çünkü saldırganlar genellikle normal ağ trafiği ve etkinlikleriyle harmanlanan teknikler kullanır. AD saldırıları arasında, AD tarafından kullanılan Kerberos kimlik doğrulama protokolündeki içsel zayıflıkları kullanan Kerberoasting saldırısı bir belirti göstermemesi nedeniyle en gizli saldırılardan biridir. Bu husus, güvenlik ekiplerinin geleneksel güvenlik araçlarını kullanarak tespit etmesini zorlaştırır. Bu çalışmada, denetimli makine öğrenimi algoritmalarını kullanarak Kerberoasting saldırılarının tespiti için bir çözüm sunmaya çalışacağız. Ayrıca, hassas verilerin güvenliğini korumak adına Kerberoasting saldırıları için makine öğrenimi algoritmalarının verimliliğini ölçmek için kullanılabilecek kamuya açık bir veri seti yoktur. Bu nedenle, çalışmayı sanal ortamda yürüterek bir veri seti oluşturduk ve güvenlik olay günlüklerini kamuya açık hale getirdik.

Anahtar Kelimeler: Aktif Dizin Güvenliği, Kerberos Protokolü, Denetimli Makina Öğrenme Algoritmaları

To My Family

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, Assoc. Prof. Dr. Cihangir TEZCAN...

TABLE OF CONTENTS

ABSTRACT iv

ÖZ v

DEDICATION vi

ACKNOWLEDGMENTS vii

TABLE OF CONTENTS viii

LIST OF TABLES xii

LIST OF FIGURES xiii

LIST OF ABBREVIATIONS xv

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation 4

1.2 Scope and Research Questions 5

2 BACKGROUND 7

2.1 Cryptographic Primitive 7

2.1.1 Encryption 7

2.1.2 Hashing 9

2.1.3 Hash-Based Message Authentication Code 10

2.2 Main Concepts of Active Directory 11
2.2.1 Introduction to Directory 11
2.2.2 Logical Framework of Active Directory 12
2.2.3 Physical Structure of Active Directory 14
2.3 Authentication and Authorization Concept 15
2.3.1 Authentication in Active Directory 16
2.3.2 NTLM Authentication 19
2.3.3 Kerberos Authentication 20
2.3.4 Common Vulnerabilities and Exposures about Kerboros Protocol 23
3 KERBEROASTING AND DETECTION METHOLOGIES 25
3.1 Understanding Kerberoasting Attack 25
3.2 Detection Techniques 28
3.2.1 Signature-Based Recognition 28
3.2.1.1 Detecting via TGS Requests with Weaker Encryption Algorithm 28
3.2.1.2 Requests for Suspicious Service Tickets 29
3.2.1.3 Using a Honeypot to Detect Kerberoasting 30
3.2.1.4 Utilizing PowerShell Logs to Identify Kerberoasting 30
3.2.2 Anomaly Based Detection 31
3.2.2.1 Statistical Methods 31
3.2.2.2 Machine Learning Methods 31
3.2.2.3 Classifier-Based Anomaly Detection 32
3.2.2.4 Identifying Anomalies With Finite State Machines 32

4 EXPERIMENTAL STUDY 33

4.1 Business Understanding 35

4.2 Data Understanding 36

4.3 Data Preparation 38

4.3.1 Data Collection 38

4.3.2 Cleaning Data 40

4.3.3 Data Transformation 41

4.3.3.1 Feature Engineering 41

4.3.3.2 Feature Selection 43

4.3.4 Data Reduction 47

4.3.5 Data Splitting 47

4.4 Modelling 48

4.4.1 Gaussian Naive Bayes 50

4.4.2 Bernoulli Naive Bayes 50

4.4.3 Decision Tree Classifier 51

4.4.4 Logistic Regression 51

4.4.5 Support Vector Classifier 51

4.4.6 Random Forest Classifier 52

4.4.7 Extreme Gradient Boosting Machine (XGBoost) 52

4.4.8 Light Gradient Boosting Machine (Light GBM) 52

4.4.9 K-Nearest Neighbors (KNN) 53

4.5 Evaluation 53

5 RESULTS 59

5.1 Discussion 59

5.2 Contributions 60

5.3 Future Work 60

5.4 Limitations 61

6 CONCLUSION 63

REFERENCES 65

APPENDICES

A CODES FOR DATA ANALYSIS 73

A.0.1 Correlation Heatmap 73

A.0.2 Explanatory Data Analysis 74

A.0.3 Feature Importance 75

B CODES FOR MODEL EVALUATION 77

LIST OF TABLES

Table 1 Allowed Encryption Types in Windows 11

Table 2 Windows Logon Types 17

Table 3 Windows Authentication Protocols 17

Table 4 Services in Lab Environment 37

Table 5 Feature Engineering 42

Table 6 Account Type of Services 42

Table 7 Ratio of Malicious Samples 48

Table 8 Hyperparameter Grid and Best Hyperparameters for Machine Learning Models 49

Table 9 Results for K-Fold Stratified Cross Validation (Average) 55

Table 10 Results for 0.012 Ratio of Malicious Samples 56

Table 11 Results for 0.008 Ratio of Malicious Samples 57

Table 12 Results for 0.004 Ratio of Malicious Samples 58

Table 13 Comparison of ML algorithms in terms of Kerberoasting Detection 59

LIST OF FIGURES

Figure 1	Implementation of Hash-Based Message Authentication Code	10
Figure 2	Logical Structure of Active Directory	13
Figure 3	Organizational Unit Structure	13
Figure 4	Replication Between Sites	15
Figure 5	Local and Domain Login Process	18
Figure 6	NTLM Authentication	19
Figure 7	Kerberos Authentication Service Request	21
Figure 8	Kerberos Authentication Service Response	21
Figure 9	Kerberos TGS Request	22
Figure 10	Kerberos TGS Response	22
Figure 11	Kerberos Application Request	23
Figure 12	Kerberoasting Attack Diagram	27
Figure 13	Weaker Encryption Algorithm in Kerberos Protocol	29
Figure 14	Offline Password Cracking	34
Figure 15	CRISP-DM Methodology	35
Figure 16	Virtual Lab Environment	37
Figure 17	Data Preparation Process	38
Figure 18	Event ID 4769 Kerberos Service Ticket Request	39
Figure 19	Service Ticket Requests of Domain	40
Figure 20	Service Ticket Requests with Weak Encryption Algorithm	41
Figure 21	Spearman Correlation Matrix Heatmap	44
Figure 22	Explanatory Data Analysis Matrix	45
Figure 23	Explanatory Data Analysis Graphs	45

Figure 24 Feature Importance Scores with All Models 46
Figure 25 Feature Importance Scores Without Light GBM Model 46
Figure 26 Kerberoasting Attack Detection Confusion Matrix 54
Figure 27 Result Graph for Test Dataset-1 with 0.012 Malicious Samples Ratio 56
Figure 28 Result Graph for Test Dataset-2 with 0.008 Malicious Samples Ratio 57
Figure 29 Result Graph for 0.004 Malicious Samples Ratio 58
Figure 30 Code for Correlation Heatmap 73
Figure 31 Code For Explanatory Data Analysis 74
Figure 32 Feature Importance Code Part-1 75
Figure 33 Feature Importance Code Part-2 76
Figure 34 Model Evaluation Code Part-1 77
Figure 35 Model Evaluation Code Part-2 78
Figure 36 Model Evaluation Code Part-3 79
Figure 37 Model Evaluation Code Part-4 80
Figure 38 Model Evaluation Code Part-5 81

LIST OF ABBREVIATIONS

ABAC	Attribute-Based Access Control
ACL	Access Control List
AD	Active Directory
AES	Advanced Encryption Standard
AI	Artifical Intelligence
API	Application Programming Interface
AS	Authentication Server
AS-REP	Authenticated Server Response
AS-REQ	Authentication Service Request
CBC	Cipher Block Chaining
CRC	Cyclic Redundancy Check
CRISP-DM	Cross Industry Standard Process for Data Mining
CVE	Common Vulnerabilities and Exposures
DES	Data Encryption Standard
DH	Diffie-Hellman
DIT	Directory Information Tree
ECC	Elliptic Curve Cryptography
EDA	Explanatory Data Analysis
FSM	Finite State Machine
FSMO	Flexible Single-Master Operator
GBM	Gradient Boosting Machine
GPO	Group Policy Objects

HMAC	Hash-Based Message Authentication Code
ICS	Industrial Control Systems
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IV	Initialization Vector
IOC	Indicators of Compromise
IPSec	Internet Protocol Security
IT	Information Technology
KDC	Key Distribution Center
KNN	K-Nearest Neighbours
LDAP	Lightweight Directory Access Protocol
LOC	Local Outlier Factor
LSA	Local Security Authority
MD5	Message Digest Algorithm 5
MFA	Multi-Factor Authentication
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MSTIC	Microsoft Threat Intelligence Center
NC	Naming Context
NTDS	Windows New Technology Directory Services
NTLM	New Technology Local Area Network Manager
OU	Organizational Unit
PCAP	Packet Capture
PKINIT	Public Key Cryptography for Initial Authentication in Kerberos
RADIUS	Remote Authentication Dial-In User Service
RBAC	Role-Based Access Control

RC4	Rivest Cipher 4
RFC	Random Forest Classifier
RFE	Recursive Feature Elimination
RSA	Rivest, Shamir, Adleman
SAM	Security Accounts Manager
SCADA	Supervisory Control and Data Acquisition
SHA	Secure Hash Algorithm
SIEM	Security Information and Event Management
SPN	Service Principal Name
SSL	Secure Sockets Layer
SSO	Single Sign On
SSPI	Security Support Provider Interface
SVC	Support Vector Classifier
TGT	Ticket Granting Ticket
TGS	Ticket Granting Service
TGS-REQ	Ticket Granting Service Request
TGS-REP	Ticket Granting Service Response
TLS	Transport Layer Security
TTP	Tactics, Techniques, and Procedure
XGBoost	Extreme Gradient Boosting Machine

CHAPTER 1

INTRODUCTION

A Directory Service is a centralized database or repository that stores, organizes, and provides access to information within a computer network. It essentially functions as a directory or catalog, holding data about network resources, such as devices, users, groups, and various other network objects [1]. Microsoft's directory service product Active Directory (AD) is the most prevalent directory service and it has the ability to centrally manage IDs throughout its infrastructure. With around 90% of Global Fortune 1000 organizations utilizing Microsoft AD as their primary method for authentication and authorization, AD is now the most widely used way to manage domain networks [2]. It is outfitted with various role services, features, and components that enables to manage identities safely and effectively in accordance with business needs. AD has been enhancing for the past 20 years, and consolidates its approach to meeting industry needs and protecting identity infrastructures from evolving security risks [3]. Given what data AD maintains and its importance, it is not unexpected that it is an appealing target for cyber attackers. Microsoft stated that over 50 million password-based attacks were occurring daily on Azure AD per day as of August 2021 [4].

In areas such as healthcare, technology, industrial control systems (ICS) automation between information technology (IT) networks has increased efficiency in all areas. A vulnerability in AD, one of the most basic and widespread technology infrastructure elements that provide connectivity between them, can have dire consequences. For instance, it can allow attackers to compromise electrical grids through Supervisory Control and Data Acquisition (SCADA) systems that manage power generation and distribution, which can cause electricity loss in countries and cities [5]. Many devastating cyberattacks have occurred and continue to occur against AD. In those attacks, one of the most potent methods employed by attackers is the Kerberoasting attack which is a type of attack where an attacker targets and extracts encrypted AD credentials from a service account to crack offline and gain unauthorized access. Kerberoasting attack serves as an arsenal for attacks against AD. So, there are striking examples of attacks linked to the Kerberoasting attack [6]. One of the attack known Solarigate was named as "one of the most sophisticated and protracted intrusion attacks of the decade" by the Microsoft 365 Defender Research Team. The 16 Tactics, Techniques and Procedures (TTP) used in the attack were highlighted by the Microsoft Threat Intelligence Center (MSTIC). Kerberoasting attack was on this list as well. The Carbon Spider attack is another instance of Kerberoasting. This attack made the ransomware available to other malicious actors, who in turn pay Carbon Spider a portion of the generated money. Among the

tools that were regularly utilized for this endeavor were CrackMapExec[1], Mimikatz[2], PowerSploit[3], and SessionGopher[4], as well as Kerberoasting. It seems that, many attacks like these examples will continue to occur in the future. So, it is also critical to understand how Kerberoasting attack occur in order to detect and mitigate future efforts.

Kerberos, is an authentication protocol that verifies the identities of users or services on a network. AD integrates with the Kerberos authentication protocol to ensure secure authentication and authorization processes within a networked environment. The client and server can encrypt their communications for privacy and data integrity assurance after they have authenticated with one another [7]. It authenticates entities by issuing tickets, providing mutual authentication between clients and servers without transmitting actual user credentials over the network. In the Kerberos protocol, passwords are not stored or transmitted in plaintext [8]. Instead, they are securely stored on the Key Distribution Center (KDC) server in a hashed form. KDC is the service in charge of principal authorization and authentication [9]. When a user sets their password or changes it, the KDC hashes the password using a one-way hash function, like MD4, MD5, or more securely, the HMAC-SHA-1 or AES. The Kerberos protocol gives clients and servers connected to insecure networks a means of verifying each other's identities using robust cryptography.

Kerberoasting is a technique that focuses on exploiting vulnerabilities from the implementation of the Kerberos protocol in Windows systems. In order to better explain this attack type, it would be better to briefly touch on the layered architecture structure in AD [10]. In domain, accounts are categorized based on sensitivity: Tier 0 is critical, including domain admins; Tier 1 related to services (web, file, sql etc.) in domain, which are implemented by user or host based service accounts, and Tier 2 has less critical ones which are normal user accounts. Attackers often target service accounts in higher tiers after breaching perimeter security using previously acquired user account credentials. These attackers aim to compromise these service accounts, which typically hold elevated privileges, enabling them to access sensitive data and critical systems. Kerberoasting leverages especially security weakness of service accounts that are used by applications or services to access network resources or to perform specific tasks without requiring interaction with an individual user. By extracting encrypted tickets granted to these service accounts, attackers aim to crack the passwords offline, allowing unauthorized access to sensitive information and resources. This method poses a serious threat to the overall security of the domain by potentially granting attackers illicit entry and control over critical systems or data [11]. The biggest reason why an attacker can do this is the weaknesses in the architecture of the Kerberos protocol implementation in AD. The primary characteristic weakness that sets Kerberos apart is its statelessness [12]. Due to this feature, it does not keep track of which account wants to access which service. Thus, the attacker can request access rights to any service, just like a normal user. Furthermore, the authentication server is unable to verify if the user making the request is authorized to view content. And also, weak or frequently used passwords might compromise Kerberos security. These passwords are hardly changed hence simple to hack. Abundance information of tickets, which

[1] An open-source post-exploitation tool called CrackMapExec (CME) helps evaluate network security. Its purpose is to make reconnaissance and exploitation in Windows-based networks easier for penetration testers and security experts.

[2] Mimikatz is a powerfull post-exploitation tool that makes it possible to extract hashes and plaintext passwords from Windows operating system memory in order to get login credentials.

[3] A set of PowerShell scripts called PowerSploit is used for post-exploitation and penetration testing purposes. It was created to help security experts with offensive security assessments in Windows settings.

[4] SessionGopher is a PowerShell program that assists in gathering and retrieving Windows Remote Desktop sessions from memory, along with user data and plaintext passwords, for post-exploitation actions.

an attacker may exploit, is another flaw. The attacker can run queries to find the service accounts in this manner. Thus, attackers can identify accounts with weak or vulnerable service principal names (SPNs)[5] [13]. An attacker must maintain or migrate inside a system. Due to the fact that it naturally protects adversaries from detection and just takes simple abilities to perform, an attackers may utilize Kerberoasting to accomplish this persistence or mobility as part of a more successful attack. These attacks can involve ransomware, the stealthy removal of data from a system, or the creation of a back door for future access.

By understanding the underlying attack vectors and developing effective detection methodologies, security teams can proactively identify and respond to potential Kerberoasting threats. Of course, research has been done before to identify one of these sneaky and hazardous Kerberoasting attack. To detect the Kerberoasting attack, Kotlaba et al. [14] mentioned two methods in their articles. One is a statistical methods and another is detection method with the help of machine learning. With the statistical methods, they implemented gathering data according to the Windows event logs on the base of KDC events. After collecting dataset, they analized False Pozitives (FP) and True Pozitives (TP). They discovered that, rules for FAR (False Alarm Ratio) may differ, particularly in settings with a variety of users. And also they utilized same methods while using non-standard techniques, such as using PowerShell monitoring or honeypots for detection. Second results were outperformed according to the rules-based service ticket requests. However, they concluded that non-standard techniques come with implementation overhead, therefore they might not be appropriate for use in all settings. In the second study which is conducted by Kotlaba et al. [15], they suggested implementing machine learning (ML) approaches, to enhance the detection of Kerberoasting attacks. They used various hyperparameter parameters and Windows event-extracted feature sets to undertake extensive experimentation with the Classifier Support Vector (SVC) [6] and Random Forest Classifer (RFC)[7] algorithms. In comparison to the statistical methods strategy, the ML approach dramatically decreased the quantity of false-positive detections, regarding to the findings. With support of the ML, they also did not see a rise in the quantity of false negative results over this time.

In the article [16] which is published in the Journal of Aeronautics and Space Technologies in 2023, a study was carried out to develop a log correlation methods against each AD attacks. With this technique, log correlation rules have been applied to strengthen the defense in depth as well as the perimetal security against attacks made in AD. Log correlation rules were developed using Splunk[8] technology. With this study, SIEM (Security Information and Event Management) managers were given information on how correlation rules can be made, regardless of environment characteristics and size.

In the study [17], a technique to categorize datasets according to attack phases as opposed to attack types is put forth by Younisse et al. They claimed that after categorizing according to attack phases,

[5] SPN stands for a unique identification that is issued to a service that is operating on a server in a Windows environment. Kerberoasting is a particularly insidious attack that targets the Kerberos protocol, which is integrated into Active Directory. It is used in the context of computer security and networking. When a client and a service need to mutually authenticate, it helps to find and authenticate the service.

[6] Support Vector Classifier is a powerful supervised machine learning algorithm used for classification tasks by finding the best hyperplane that separates different classes in data, maximizing the margin between them for accurate predictions in complex datasets.

[7] A combined machine learning approach called a Random Forest Classifier uses several decision trees to decrease overfitting and increase classification accuracy.

[8] Splunk is a powerful data analytics platform that enables organizations to collect, index, search, and analyze machine-generated data from various sources in real-time.

these datasets can be used to build clever defense models in machine learning models. This study also introduces an approach to predict and identify any types of AD attacks. The main purpose of the study is to label datasets for SIEM products to be used in machine learning. Younisse et al. claims that after being categorized according to attack stages, these data sets can be used to create intelligent defense models in machine learning models. In their work, the stages of the Kerberoasting attack were studied and they collected data about attack scenario. Data in pcap (Packet Capture) files is used to highlight features and specific actions at each stage of the attack lifecycle. Each pcap file provided data for a different phase of the attack lifecycle. In light of this information, they recommended that it can be used to train ML and Artificial Intelligence (AI) models to create a strong defense against attacks.

This thesis aims to explore the technical details of Kerberoasting attacks within AD environments, focusing specifically on the detection methods employed to identify and mitigate such attacks. To detect Kerberoasting attack with statistical methods, the false positive ratio could not determined clearly, due to the dynamic parameters of the environment specific to itself. Then with the second study, false positive alarm ratio has been reduced to zero with the help of one class SVC and RFC machine learning algorithms. However, it was observed that the false negative value was still high in those algorithms. Since an alarm for a non-existent attack is much better than no alarm for an existing attack, we investigated in this study, whether there are other supervised machine learning algorithms that provide better results. Because missing a Kerberoasting attack can cause domain compromise. Therefore, false negative results might have dire consequences in terms of Kerberoasting attack. Supervised machine learning algorithms were chosen in the study due to fact that each IT environment has its own characteristics and the algorithm detects anomalies in the environment as soon as possible without wasting time with pre-trained model. Thus, with the method shown in detail, there will be a detailed Kerberoasting attack detection guide with machine learning that can be used by security personnel.

1.1 Motivation

What makes Kerberoasting particularly concerning is its stealthy nature. The attack is conducted offline, allowing adversaries to crack passwords associated with targeted service accounts without raising immediate alarms. As a result, organizations may remain unaware of the breach until after the damage has been done, leading to severe consequences such as data breaches, financial losses, and reputational damage.

The need for effective detection methodologies for Kerberoasting attacks is significant. By promptly identifying and mitigating such attacks, organizations can significantly reduce the risk of unauthorized access and data compromise. However, current detection approaches often fall short in adequately identifying these stealthy attacks. Traditional security measures, such as signature-based recognition or rule-based systems, are ill-equipped to detect Kerberoasting due to its offline nature and the absence of traditional indicators of compromise (IOCs). As a result, organizations find themselves exposed to this insidious threat, with limited visibility into potential breaches and compromised service accounts.

The increasing prevalence and stealthiness of Kerberoasting attacks present a clear and pressing need for effective detection methodologies within Active Directory environments. By addressing this gap in knowledge and proposing practical solutions, this research aims to empower organizations to detect and mitigate Kerberoasting attacks promptly, thus protecting their critical assets and maintaining the trust of their stakeholders.

1.2 Scope and Research Questions

The scope of the thesis is going to focus on the detection methodologies for Kerberoasting attacks within AD environments. The research is going to involve investigating various existing detection techniques, exploring their strengths and limitations, and proposing novel approaches to enhance the detection capabilities. The scope will encompass:

- Examining the technical details of Kerberoasting attacks, including the underlying vulnerabilities in the Kerberos authentication protocol and the methods employed by attackers.
- Conducting experiments and simulations to evaluate the accuracy of the proposed detection methodologies.

Throughout the thesis the research questions are as follows:

- What are the key vulnerabilities in the Kerberos authentication protocol that make it susceptible to Kerberoasting attacks?
- How do attackers identify and target service accounts with weak or vulnerable service principal names (SPNs)?
- What are the strengths and limitations of different detection methodologies in terms of accuracy, efficiency, and scalability?
- Is there a machine learning algorithm that minimizes false negative alarms along with false positive alarms?
- No publicly available dataset, how can we create a dataset in a scientific way in term of Kerberoasting attack?

CHAPTER 2

BACKGROUND

This section provides to the reader a technical knowledge base in order to better understand the Kerberoasting attacks. Furthermore, through the Background section it can be found fundemental information regarding Cryptographic Primitive, Main Concepts of Active Directory as well as Authentication and Authorization concepts. These topics are explained briefly in align with thesis subject area.

2.1 Cryptographic Primitive

An essential component of electronic key systems is cryptography. It is utilized for access control, digital signatures, data secrecy, and other purposes [18]. Cryptography is the practice and study of techniques used to secure communication and information. It plays a crucial role in ensuring the confidentiality, integrity, and authenticity of data in various digital systems. Cryptography also plays a pivotal role in the realm of authentication within IT systems. It ensures secure verification of identities by employing encryption and hashing techniques to protect sensitive information during login procedures. Through encryption, authentication data such as login credentials is protected from unauthorized access, while hashing function enables secure storage and validation of passwords without storing them in plaintext, thereby fortifying the authentication process against potential security breaches. In order to have a clearer understanding of the concepts that we will frequently encounter in the subjects of authentication and authorization, the concepts of encryption and hashing function are explained concisely.

2.1.1 Encryption

Encryption is a process used to convert readable and understandable data, often referred to as plaintext, into an encoded and scrambled form known as ciphertext. It involves using algorithms and keys to transform information in such a way that only authorized parties possessing the appropriate key can decipher and understand the encrypted data. The primary purpose of encryption is to ensure confidentiality and privacy by securing sensitive information from unauthorized access or interception [19]. It prevents unauthorized individuals or entities from understanding the content of the encrypted data, thus preserving it against potential breaches or data theft.

In terms of authentication concept we are going to mention symmetric and asymmetric cryptography briefly.

Symmetric Cryptography: This type of encryption uses the same key for both the encryption and decryption processes. The sender and receiver need to share this secret key beforehand to secure their communication. Examples of symmetric encryption algorithms include DES (Data Encryption Standard), RC4 (Rivest Cipher 4) and AES (Advanced Encryption Standard).

- **DES :** One of the earliest symmetric encryption standards. However, due to its key length of only 56 bits, which is now considered too short to provide adequate security against modern threats, DES has largely been replaced by more secure algorithms like AES. With current technology and computational resources, it is feasible to break DES through exhaustive search methods within a reasonable amount of time [20].
- **RC4 :** A stream cipher known for its simplicity and speed. RC4 is adaptable to various security requirements since it provides varied key sizes ranging from 40 bits to 2048 bits. It was widely used in various applications, including wireless networks and secure web browsing, but vulnerabilities have been discovered in some implementations, leading to its decreased usage. Internet Engineering Task Force (IETF), have discouraged the use of RC4 in secure communication protocols and applications. Major web browsers and other software have also deprecated or removed support for RC4 due to its weaknesses [21].
- **AES :** It is one of the most widely used symmetric encryption algorithms. It operates on fixed block sizes of data and supports key sizes of 128, 192, or 256 bits. AES has been adopted by governments and organizations worldwide due to its strong security and efficiency.

Asymmetric Cryptography: Also known as public-key cryptography, asymmetric cryptography uses a pair of keys: a public key and a private key. The public key is used for encryption, while the private key is used for decryption. Messages encrypted with the public key can only be decrypted using the corresponding private key, which is kept confidential. This method is often used for secure communication, digital signatures, and key exchange. Examples of asymmetric encryption algorithms include RSA (Rivest, Shamir, Adleman), ECC (Elliptic Curve Cryptography), and DH (Diffie-Hellman).

- **RSA :** The security of the RSA public key encryption technique is based on the factoring of integers being an algorithmic challenge [22]. It named after the inventors of this widely used asymmetric encryption algorithm, which relies on the mathematical complexity of prime factorization for secure communication. It uses a public key for encryption and digital signatures, and a corresponding private key for decryption and signature verification.
- **ECC :** Based on the algebraic topology of elliptic curves over finite fields, ECC is a public key cryptography system [23]. The primary characteristic that sets Elliptic Curve Cryptology apart from the others is its requirement for a lower key size than other encryption techniques.
- **DH :** A unique technique for exchanging cryptographic keys is called Diffie-Hellman Key Exchange. Through the use of insecure media, two parties can acquire a shared secret key using the DH key exchange technique [24].

Overall, encryption ensures data security, confidentiality, and integrity, making it a fundamental aspect of modern-day cybersecurity practices and information protection in various digital systems and communication channels.

2.1.2 Hashing

Hashing is a process used to convert any input data, such as a message, file, or piece of text, into a fixed-size string of characters through a mathematical algorithm [25]. The output generated by this algorithm is called a hash or digest.

Unlike encryption, hashing is a one-way function, meaning it is designed to be irreversible. It produces a unique output for each input, and even a small change in the input data results in a significantly different hash value. Hashing is similar to encryption, the only difference between hashing and encryption is that hashing is one-way, meaning once the data is hashed, the resulting hash digest cannot be cracked, unless a brute force attack is used [26]. The primary purposes of hashing includes [27];

Data Integrity Verification: Hash functions are used to generate checksums or hash values for files or data. Comparing the computed hash of a file to a previously stored hash allows verification of the file's integrity. Any alteration in the file content would result in a different hash value.

Keeping Passwords Secure: In authentication systems, instead of storing actual passwords, systems store their hashed versions. When a user enters a password during authentication, the system hashes it and compares the resulting hash to the stored hash. This method secures passwords against exposure even if the stored hashes are compromised.

Commonly used hash functions include Secure Hash Algorithm (SHA) variants like SHA-256, SHA-1, and SHA-3, as well as Message Digest Algorithm 5 (MD5). These hash functions are essential in various security applications, ensuring data integrity, password protection, and secure authentication mechanisms.

- **MD5 :** This cryptographic hash function technique converts an input message of arbitrary length into a fixed-length message of 16 bytes. The message-digest algorithm is known by its acronym, MD5. With the goal of enhancing security, MD5 was created as an upgrade to MD4. MD5 always produces an output of 128 bits (digest size) [28]. MD5 is no longer considered secure for cryptographic purposes due to several vulnerabilities and weaknesses that have been identified in the algorithm. Collision attacks, where different inputs produce the same hash value, are among the significant vulnerabilities found in MD5 [29].

- **SHA :** Secure hashing algorithm is known by its acronym, SHA. A modified version of MD5, called SHA, is used to hash data and certificates [30]. An incomprehensible smaller version of the original data is produced by a hashing algorithm by the use of bitwise operations, modular additions, and compression functions. There are multiple hash functions with different hash lengths in the SHA (Secure Hash Algorithm) family. The following are a few popular SHA algorithms:

  SHA-1, a once widely used cryptographic hash function, produces a 160-bit hash value, but it is now considered insecure due to vulnerabilities allowing collision attacks, and its use is strongly discouraged in favor of more secure alternatives like SHA-256.

  SHA-2, a family of cryptographic hash functions that includes SHA-224, SHA-256, SHA-384, SHA-512, and others, is widely adopted for its secure and efficient generation of hash values, with longer bit lengths providing increased resistance against collision attacks compared to its predecessor, SHA-1.

SHA-3, these hash functions based on a different internal structure known as the Keccak sponge construction, provides an alternative to SHA-2 with potential advantages in security and flexibility, though it has seen less widespread adoption in practical applications.

2.1.3 Hash-Based Message Authentication Code

It is a type of authentication code that involves a cryptographic hash function used in combination with a secret key known as hash-based message authentication code, or HMAC. HMAC is designed to verify both the integrity and authenticity of a message or data. A hash function and a secret key are two components of the cryptographic authentication method [31].

The purpose of HMAC is to ensure that data has not been altered or tampered with during transmission and to verify the identity of the sender. It uses a cryptographic hash function (such as MD5, SHA-1, SHA-256, etc.) and a secret key to generate a hash-based message authentication code. As seen in the Figure 1, HMAC algorithm works by taking the message or data input and combining it with the secret key. It then processes this combined input through the chosen hash function to generate a unique hash value. The resulting hash, known as the HMAC, serves as a tag or fingerprint for the message. The recipient of the message or data can independently compute the HMAC using the same hash function and secret key. If the computed HMAC matches the one received with the message, it verifies that the message has not been altered in transit and that the sender possesses the correct secret key. Any modification to the message or a different key would result in a different HMAC value, thereby indicating potential tampering or unauthorized access.

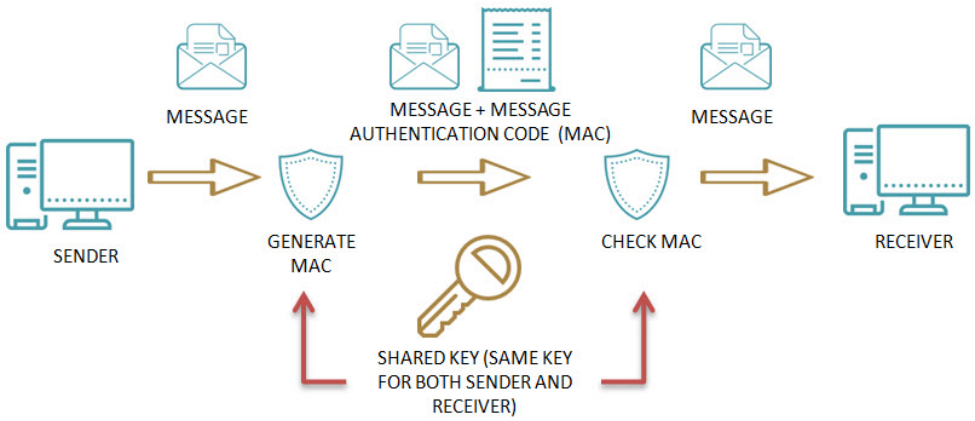

Figure 1: Implementation of Hash-Based Message Authentication Code

HMAC is commonly used in various secure communication protocols, including secure sockets layer (SSL), transport layer security (TLS), and internet protocol security (IPsec), as well as in authentication mechanisms for APIs, digital signatures, and other applications where data integrity and authentication are essential. The key used in HMAC needs to be a specific length to work with the hash function. If the key is shorter than the block size of the hash function, it is padded (or adapted) to meet the required length. This process of adjusting the key to the right length involves XORing parts of the key with specific constants and concatenating the results. It ensures that the key is suitable for secure processing within the HMAC algorithm.

The encryption types that Kerberos uses in Windows operating systems are displayed in Table 1 [32]. Suites using the AES cipher have been established as the default starting with Windows Server 2008 and Windows Vista, replacing the earlier suites with the RC4 cipher. Additionally, as with Windows 7 and Windows Server 2008 R2, cipher suites utilizing the DES cipher have been blocked. Therefore, for security, RC4 and DES encryption algorithms should not be used, AES encryption algorithm should be preferred.

Table 1: Allowed Encryption Types in Windows

Cipher Suite Name	**Type**
DES_CBC_CRC	0x1
DES_CBC_MD5	0x3
RC4_HMAC_MD5	0x17
AES128_HMAC_SHA1	0x11
AES256_HMAC_SHA1	0x12

CBC (Cipher Block Chaining): CBC is a mode of operation for block ciphers like DES. In CBC mode, each plaintext block is combined with the previous ciphertext block before being encrypted. It introduces an initialization vector (IV) to prevent patterns in the plaintext from being visible in the ciphertext.

CRC (Cyclic Redundancy Check): CRC stands for the integrity-checking mechanism. It is used to detect changes or alterations in the data during transmission. CRC is a type of hash function used to produce a checksum, which is appended to the data before encryption. Upon decryption, the checksum is recalculated and compared to the decrypted data to ensure integrity. CRC is a non-cryptographic method used for error detection. It generates a checksum based on the data being checked. The checksum is appended to the data and recalculated at the receiver's end to verify if any errors occurred during transmission. CRC is not designed for security but rather for detecting accidental changes in data.

Therefore, "DES-CBC-CRC" implies that the DES encryption is using CBC mode for encryption and a CRC-based integrity check to detect any potential tampering or corruption of the encrypted data during transmission.

2.2 Main Concepts of Active Directory

In order to be familiar with Active Directory, the concepts of Introduction Active Directory, Logical and Physical Framework of Active Directory and Authentication and Authorization are going to be explained respectively in the chapter.

2.2.1 Introduction to Directory

A directory service is a database-based tool used to store and manage user, resource information. Directory services are used by network and system administrators to create new users, adjust user rights,

and keep an eye on and regulate access to infrastructure resources and applications. An application of service has to verify that a client is authentic and has the necessary rights to access and use it by consulting the directory service when the client reaches it [33].

Directory service implementation is governed by standards. Several directory services beyond AD are utilized across IT environments. These encompass LDAP[1] (Lightweight Directory Access Protocol), a foundational protocol for accessing and managing distributed directory information services over IP networks. The evolution of Directory Access Protocol version X.500 turns into Lightweight Directory Access Protocol (LDAP) by the early of the 2000s. An upgraded form of LDAP, known as LDAPv3, serves as the foundation for Microsoft Active Directory. Microsoft AD has been a core component of Windows systems. OpenLDAP, an open-source implementation of LDAP, serves as a free, open-source directory service enabling centralized management of user accounts and authentication [34]. Additionally, eDirectory (formerly Novell Directory Services) by NetIQ offers robust directory-based services for identity management in enterprises. FreeIPA, integrating LDAP, Kerberos, DNS, and more, provides centralized identity management for Unix-like systems. Other notable services include 389 Directory Server, Samba Directory for Windows interoperability, and OpenDJ by ForgeRock. Each of these directory services offers distinct features and functionalities, appealing to various identity management needs in enterprise, government, and educational settings.

2.2.2 Logical Framework of Active Directory

AD logical structure is a hierarchical arrangement of objects like domains, organizational units (OUs), trees, and forests, organizing and managing network resources and users in a Windows environment. NTDS[2] database is created with DIT [3] file once Active Directory Domain Services is deployed on Windows servers. This database contains all of the items, including users, computers, servers and so on. The infrastructure of Active Directory consists of domains, trees, forests, and other components. While installing the server role for Active Directory Domain Services, a domain is established. A domain is an area within which authority is limited. In Figure 2, there is example of domains; such as eu.csec.local and usa.csec.local. To use network resources of these domains, a user has to log in these domains. There are two mostly used methods to login domain. One method is to type on login screen "username@domain name", another is "NetBIOS name\username". As seen in the examples, there are authority borders which are represented by triangle. And also there is a forest which is consisted of one root domain and two child domains represented by a circle.

[1] Lightweight directory access protocol (LDAP) is a protocol that helps users find data about organizations, persons, and more. LDAP has two main goals: to store data in the LDAP directory and authenticate users to access the directory.

[2] NTDS (Windows NT Directory Services) is the directory services used by Microsoft Windows NT to locate, manage, and organize network resources. The NTDS.dit file is a database that stores the Active Directory data (including users, groups, security descriptors and password hashes).

[3] Information about user objects, groups, and group membership is stored in the DIT (Directory Information Tree) file, an Active Directory database. The file is significant since it also contains the hashes of every user's password within the domain.

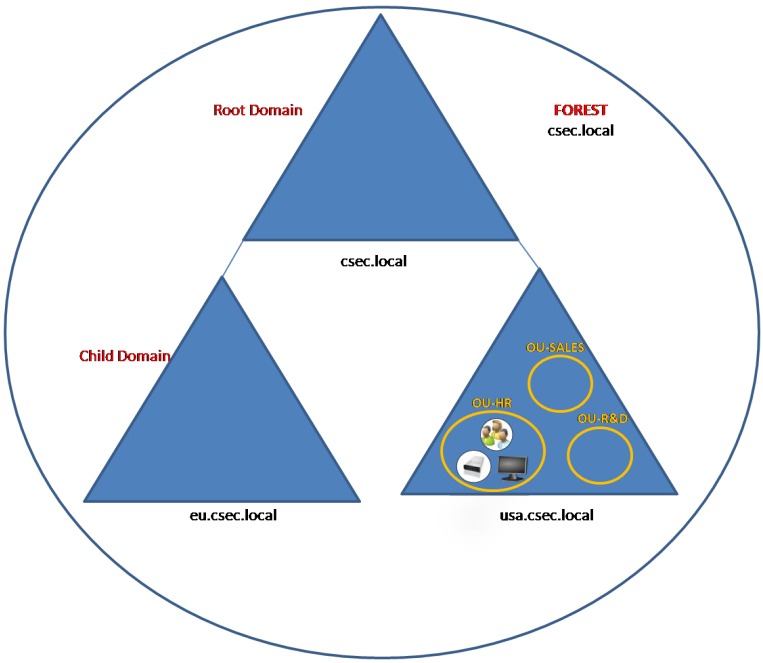

Figure 2: Logical Structure of Active Directory

Within the domain, organizational units provides smaller-scale grouping of items. For example, lets assume the sales department in a company as an example. This sales department will have a certain number of users with common features. As shown in the Figure 3, users in the sales department can be grouped logically by considering common geographic or functional features [35]. For instance, Domain admins can provide authority over such items to people or organizations in the regional offices by creating OU based on geographic regions. Additionally, rather of implementing a unique security policy for the whole sales department across all branch offices, it can be implemented at the appropriate unit level to a sales department that operates regionally. The permissions that are automatically assigned to the parent OU are passed down to all child OUs.

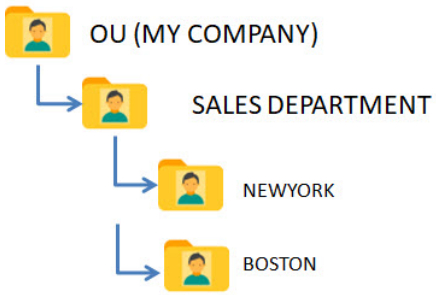

Figure 3: Organizational Unit Structure

Objects with comparable administrative and security requirements are often grouped together. Common services and resources are used by the sales department. They have comparable security needs for networks and data. Thus, we may organize all of the users of the sales department into an organi-

zational unit named sales. Rather than implementing security policies at the user level, system admin can do security policies at the OU level with GPOs (Group Policy Objects)[4].

2.2.3 Physical Structure of Active Directory

The logical and physical components are autonomous, although they play equal roles in the architecture of Active Directory Domain Services. The fundamental component of Active Directory Domain Services is replication. Any modifications made to one domain controller on a system should be reflected in the others. Logical components are more easily rearranged than physical ones [35].

Domain controllers are main components of Active Directory and they hold role of Domain Services. It may be a virtual machine or a real server. The directory partition which is a logical division of the AD database that stores specific sets of directory data, such as domain data or application data that is to be replicated to the other domain controllers within the same domain is stored on the domain controller. There can be any number of domain controllers for the domain. The size, location, and network segmentation of the company all affect how many domain controllers are needed. All other domain controllers will duplicate any object-level modifications performed in one domain controller. Nonetheless, the Flexible single-master operator (FSMO)[5] role owner is the only one authorized to make modifications to certain operational roles connected to Active Directory [36].

Partitioning data is required because of the dispersed nature of AD. Every domain controller in a forest which is a networked collection of numerous domain trees with a shared schema, configuration, and global catalog [37] would need to duplicate all of the data if data partitions were not utilized. Data should frequently be grouped according to political or geographic constraints. A domain, also known as a naming context (NC), is a huge data partition. All of the data inside a domain only has to be replicated by domain controllers that hold authoritative status. On such domain controllers, information about other domains is not required. However, certain Active Directory data has to be copied across to every domain controller in a forest. Within AD, there are three predetermined naming contexts [38];

1. *Domain naming context for each domain,*
2. *The Configuration naming context for the forest,*
3. *The Schema naming context for the forest.*

Data about the configuration, including objects that reflect LDAP rules, sites, subnets, and naming contexts, is stored in the Configuration NC. The set of object class and attribute definitions for the many kinds of data that may be saved in AD is included in the Schema NC. Domain NC, which includes information unique to a domain such as people, groups, machines, etc., exists for every domain in a forest.

The global catalog server stores a partial copy of the items in other domains within the same forest as well as the complete readable copy of the objects in its host domain. All of the forest's objects

[4] In a Windows operating system environment, Group Policy Objects (GPOs) are a collection of settings that define how computers, user accounts, and groups can operate within an Active Directory environment. GPOs are used to configure various settings and restrictions for users and computers

[5] Active Directory FSMO roles: Schema Master, Domain Naming Master, Relative ID (RID) Master, Primary Domain Controller (PDC) Emulator, Infrastructure Master. One DC may be allocated to more than one FSMO duty, or perhaps all five.

as well as the most often used query attributes are duplicated in the partial replica. Applications and users inside a domain can utilize the global catalog server to query for items located in another domain (within the same forest). Global catalog servers will not be the default setting for any domain controller inside the domain. The global catalog server is the initial domain controller that is installed; based on the needs of the company, further domain controllers may be promoted to become global catalog servers. Not all domain controllers within the domain must be a global catalog server.

The network's physical topology is defined by the Active Directory sites. Sites might be distinct buildings connected by a campus network, or they can be different cities or even nations with a branch office. Physical network segmentation is not actually taken into account in the logical architecture of Active Directory. To have a healthy identity architecture, domain controllers should replicate changes to each other as they are in the same domain and forest as shown in Figure 4.

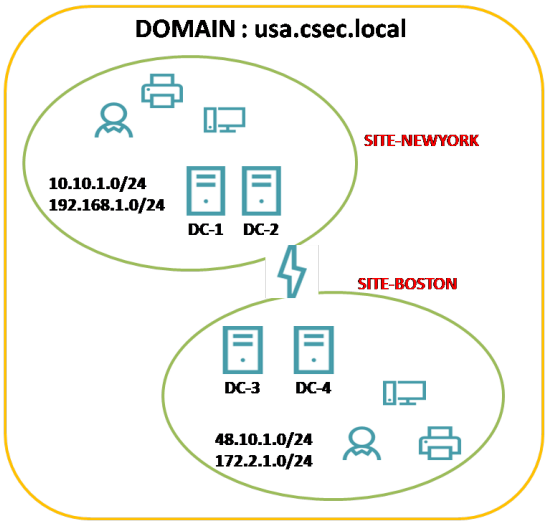

Figure 4: Replication Between Sites

2.3 Authentication and Authorization Concept

Verifying the identification of someone or something is the process of authentication. If something or someone is indeed who or what it is claimed to be, it makes this determination using data that has been given to the authenticator [39]. It is a fundamental aspect of security and is used extensively in various fields, such as computer security, online transactions, access control systems, and more. Presenting credentials is a common step in the authentication process. Additional factors like fourth and fifth factor authentication can aid in user identification and protect sensitive data. These credentials may include [40] :

1. **Something you know :** Password, PIN, or passphrase.
2. **Something you have :** Physical token, a keycard, or a mobile device (two-factor authentication).
3. **Something you are :** Biometric data, such as fingerprints, retina scans, facial recognition (three-factor authentication).
4. **Where you are :** Position information (four-factor authentication).

5. **When you are :** Time information (five factor-authentication).

To improve security, multi-factor authentication (MFA) combines two or more of these authentication elements. For example, a popular method combines a temporary code given to a mobile device (something you have) with a password (something you know). By limiting access to sensitive data, resources, or services to only authorized people or systems, authentication serves the primary purpose of preventing unauthorized access. Several sectors, such as user login systems, network security, banking, government services, and several more areas where preserving data and resources is crucial, require authentication procedures.

Determining whether an entity (a device or a person) may access resources, such as read or write data, run programs, is the process of authorization. Denial or revocation of access is another aspect of authorization, particularly for someone or something malevolent [41]. Authorization may be defined as the process of approving or rejecting access rights and permissions to verified people or entities according to their identification and the resources they are attempting to utilize. An authenticated entity's ability to conduct certain activities or processes within a system or on certain resources is determined by authorization, whereas authentication confirms a user's or system's identity. Authorization methods establish and maintain a user's or entity's degree of access, defining what they can do and what resources they can access once they have been authenticated. Establishing guidelines, regulations, or permissions that specify who has access to what information and what activities they may take is known as authorization. There are other ways to implement authorization [42], such as:

1. **Access Control Lists (ACLs):** These lists indicate which individuals or groups are authorized to utilize particular resources or carry out particular tasks.
2. **Role-Based Access Control (RBAC):** Roles are allocated to users, and these roles have related rights. The roles that are assigned to users carry over their permissions.
3. **Attribute-Based Access Control (ABAC):** Decisions about access are made in light of the environment, the resource, and the user's characteristics. When determining permission, this dynamic model takes into account a number of factors.

With authorization, you can make sure that users or systems only have access to the rights needed for the roles or responsibilities they are assigned. It is a crucial part of system and application security because it lowers the possibility of data breaches, unauthorized alterations, or resource misuse by preventing unauthorized access to critical information or functionality.

2.3.1 Authentication in Active Directory

AD ensures secure access to network resources, and maintaining the overall integrity and security of an organization's IT infrastructure. To access this functionality, an entity have to logon. Entities are authenticated to Windows-based computers by a logon process. It might be either domain-wide or local. Regardless of the authentication protocol or authenticator used, all authentications on Windows-based PCs are handled as one of numerous logon types [43]. Table 2 lists the Windows logon types. To briefly mention the important logon types; interactive logon involves a user directly accessing a computer system by providing credentials (username, password, etc.) through the system's interface, either locally or remotely, to gain immediate access to its resources. Network logon occurs when a user accesses resources, such as files or applications, hosted on a network-connected server or device. Authentication happens against a central server, allowing access to network resources based on user

permissions and security settings. Service logon involves a Windows service or application accessing resources autonomously using a designated service account without direct user involvement. It enables background operations and resource access based on permissions assigned to the service account within the system or network. To provide network login functionality, Windows systems generally use

Table 2: Windows Logon Types

Logon Type	**Type Nu.**	**Authenticators Accepted**
Interactive (also known as, Logon locally)	2	Password, Smartcard, other
Network	3	Password, NT Hash, Kerberos ticket
Batch	4	Password (stored as LSA secret)
Service	5	Password (stored as LSA secret)
Network Cleartext	8	Password
New Credentials	9	Password
Remote Interactive	10	Password, Smartcard, other

the authentication mechanisms listed in Table 3. Additionally, Table 3 lists the encryption algorithms used in these mechanisms and the attack methods on these protocols. There are also other protocols such as Digest and Schannel used in Windows systems, but these protocols are used mostly in web systems, not for logging into the domain [44]. There are two logon methods by which users can interactively log in to a computer: Local account and domain account. A local account is a user account stored on a single device, providing access and permissions only on that particular computer or device. It is independent of network domains and does not rely on external servers for authentication. Domain account is a type of account that managed and stored centrally on a network server within a Windows domain, allowing access to various resources across multiple computers connected to that domain. Domain login not only grants the user access to domain resources, but also local resources. Before explaining the process of local or domain authentication in Windows systems, it would be ap-

Table 3: Windows Authentication Protocols

Protocol	**Authentication**	**Cryptographic Algorithm**	**Attacks**
NTLMv1	Challenge-response	DES	Authentication LDAP Relay Pass-The-Hash
NTLMv2	Challenge-response	HMAC-MD5	Pass-The-Hash NS Poisoning SMB Relay
Kerberos	Third-party ticket	DES_CBC_CRC, DES_CBC_MD5 RC4_HMAC_MD5 AES128_HMAC_SHA1 AES256_HMAC_SHA1	Pass-the-ticket, As-Rep roasting Golden Ticket, DC Shadow Silver Ticket, Encryption Downgrade, Credential stuffing Brute Force, Kerberoasting

propriate to explain some concepts. These concepts are Local Security Authority (LSA) and Security Support Provider Interface (SSPI). The LSA is a component in Microsoft Windows operating systems responsible for enforcing the security policy on the system. LSA maintains security policies, validates credentials of users during logins (authentication), manages access tokens, handles security policies

like password policies, and interacts with the Security Accounts Manager (SAM) database that stores user account information. SSPI is an API in Windows that provides a standardized interface for authentication services. It allows applications to utilize various security protocols, including NTLM (NT LAN Manager) and Kerberos, without needing to understand the specific implementation details, enabling seamless authentication and security services integration within Windows-based systems. SSPI abstracts the complexities of security protocols, offering a unified way for applications to interact with security mechanisms and perform authentication, encryption, and other security-related functions [45].

For a user to log in locally, they must have an account in the Security Accounts Manager (SAM) on their local computer. Security accounts, which are kept in the local computer registry, are a way for the SAM to manage and safeguard user and group data. Network connectivity is not necessary, although it is possible for the computer. Access to local resources is controlled using local user accounts and group membership data. Figure 5 shows overview of the local authentication process indicated by black arrows.

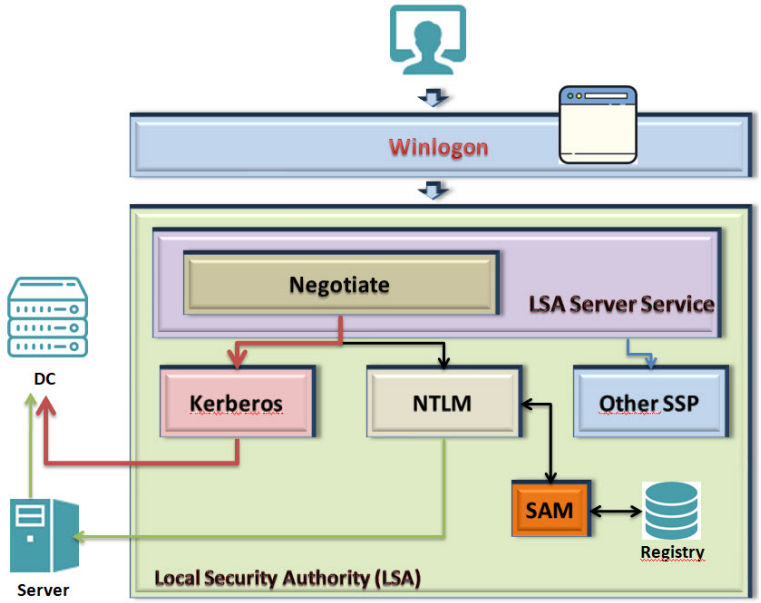

Figure 5: Local and Domain Login Process

In AD environment, entities can be user, computer and service. Within hierarchical structure, objects are identified by globally unique identifiers (GUIDs) and security Identifiers (SIDs) to distinguish between entities and manage their relationships and permissions accurately. GUIDs and SIDs are both unique identifiers used in Windows environments, but their purposes and scopes of uniqueness differ significantly. GUIDs focus on object identification across environments, while SIDs are essential for managing access control and security within a specific domain or system. Every AD item has a GUID, a 128-bit identifier that is assigned to it by the system at creation. Moving or altering objects has no effect on its worth. Although it has a very low likelihood of being replicated, GUID is not guaranteed to be unique. A variable-length identifier called a Security Identifier (SID) is used to identify a security principal. It is distinct in the field. The SID values linked to the item will alter if it is moved to a different domain. Furthermore, a domain's SID value cannot be accepted by another domain [46]. The

process indicated by the red and green arrows in Figure 5 illustrates overview the Windows domain-wide authentication process.

The Kerberos protocol is the recommended protocol to use within the AD domain as it offers more security than NTLM. NTLM is still supported, though. Instead of using Kerberos and NTLM SSPs directly, it uses the Negotiate Security Package, which makes an automated choice between the two. By default, Kerberos is chosen unless one of the systems participating in the authentication procedure is unable to utilize it. Network login is used to verify the user's identity to the network service they are trying to access after interactive logon. Since previously created credentials are reused, this is typically undetectable to the user [15].

2.3.2 NTLM Authentication

NTLM (NT LAN Manager) is an authentication protocol developed by Microsoft. It is an older authentication protocol that has been a fundamental part of Windows-based systems for a long time. It has evolved through various versions (NTLMv1 and NTLMv2) to address security weaknesses found in earlier versions.

When a user tries to access a resource on a Windows server, the server generates a random value known as a challenge and sends it to the client. The client then encrypts this challenge using a hash derived from the user's password. This cryptographic response is sent back to the server.

Upon receiving the response, the server performs its own computation based on the stored hash of the user's password to generate an expected response. It compares this expected response with the one received from the client. If the two match, the user is authenticated, and access to the requested resource is granted. In Figure 6, the authentication procedure with the NTLM authentication protocol is explained step by step.

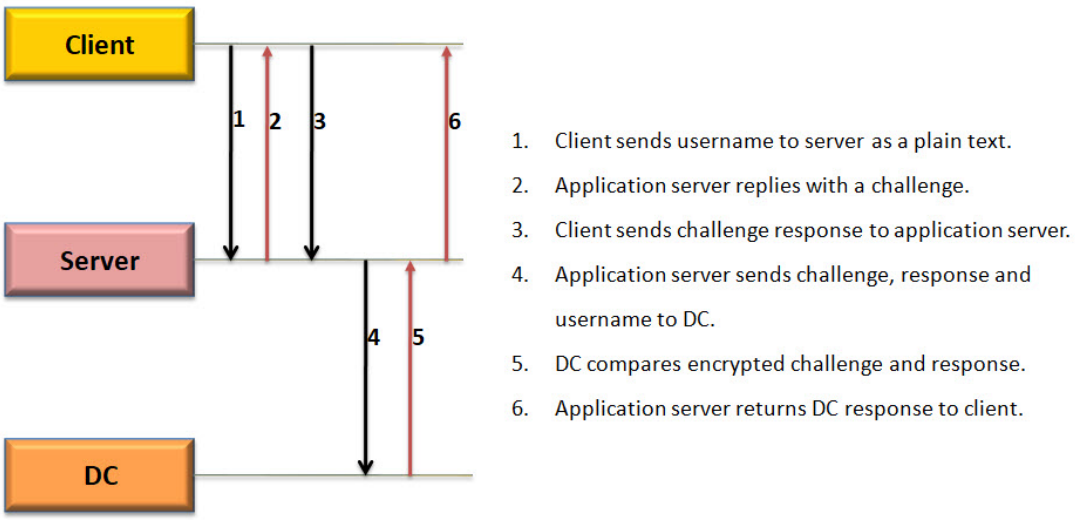

Figure 6: NTLM Authentication

For Windows authentication on systems that are set up as workgroup members, NTLM authentication is still required and supported. On non-domain controllers, NTLM authentication is also utilized for local login authentication. For Active Directory systems, Kerberos Version 5 is the recommended way; nevertheless, non-Microsoft or Microsoft applications may still utilize NTLM [47].

NTLM does not transmit the actual user password over the network; instead, it uses hashed values in the authentication process. However, NTLM has known security weaknesses, such as susceptibility to certain attacks like pass-the-hash attacks, where an attacker could potentially reuse captured hashed credentials to gain unauthorized access. Due to these security concerns, modern Windows environments often prefer more secure protocols like Kerberos for authentication purposes.

2.3.3 Kerberos Authentication

The Massachusetts Institute of Technology (MIT) designed Kerberos as a network authentication system to solve vulnerabilities with network security. The Kerberos protocol gives clients and servers connected to insecure networks the ability of verifying each other's identities using robust cryptography. The client and server can additionally encrypt their communications for privacy and data integrity assurance after they have authenticated with each other [48]. Kerberos may remind us, the concept of RADIUS (Remote Authenatication Dial-In User Service). Kerberos is a secure network authentication protocol used in distributed environments, employing tickets and a Key Distribution Center (KDC) for user authentication without transmitting passwords. In contrast, RADIUS is a protocol facilitating centralized authentication, authorization, and accounting for remote users accessing network resources through a network access server, typically used in scenarios like dial-up, VPN, or wireless networks. In short, RADIUS is a one-way authentication mechanism while Kerberos is bi-directional. RADIUS, a legacy authentication protocol, is also still subject to attacks. The Blast-RADIUS (CVE-2024-3596) attack, which was discovered on July 18, 2024, allows an attacker who mediates between a RADIUS client and a server to craft a valid protocol accept message in response to a failed authentication request [49].

To securely authenticate users and services in a network environment, there are several phases in the Kerberos authentication process. Kerberos functions in a Windows domain through a sequence of exchanges between the target service, KDC and the client. A brief explanation of the Kerberos authentication procedure in AD is provided below [50]:

1. **Request for Ticket-Granting Ticket (TGT):** A client initiates the authentication process by sending authentication requests to the KDC. This request typically occurs when the user attempts to log in to the domain or access a network resource. The client sends an AS-REQ (Authentication Service Request) message to the Authentication Service (AS) in the KDC. This message includes the user's identity (principal) and is encrypted using the user's password as the key. Figure 7 shows what information is sent during the AS-REQ process.

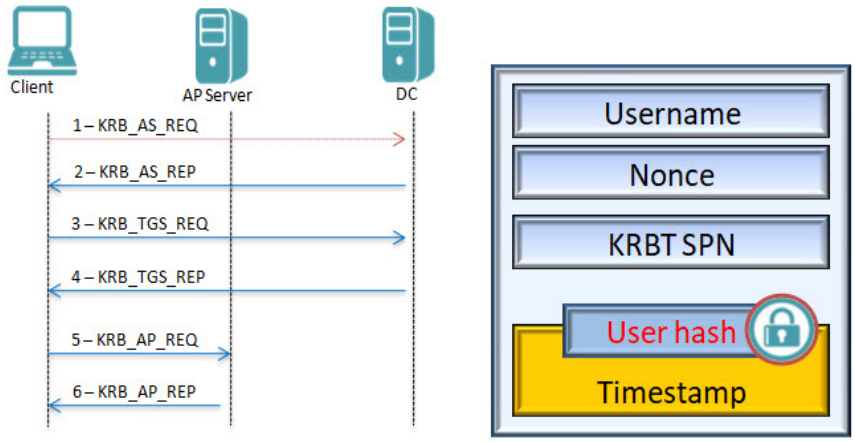

Figure 7: Kerberos Authentication Service Request

2. **Authentication Service Response:** The KDC's AS checks the user's credentials by verifying the encrypted request. If the credentials are valid, the KDC generates a TGT and sends it back to the client. The KDC sends an AS-REP (Authenticated Server Response) message containing the TGT encrypted with a session key derived from the user's password. The client receives the TGT. As seen in Figure 8, TGT is encrypted with the krbtgt account hash that is responsible for KDC service.

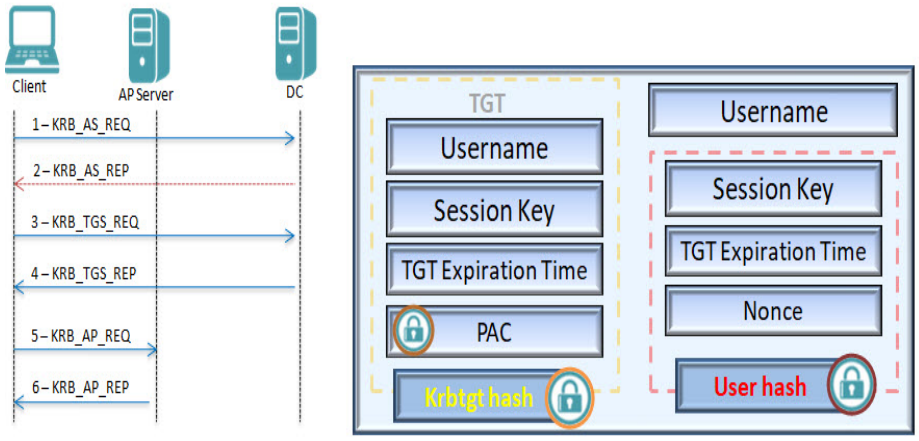

Figure 8: Kerberos Authentication Service Response

3. **Request for Service Ticket:** When the client wants to access a specific service (e.g., file server, printer), they send a request for a Service Ticket to the Ticket Granting Service (TGS) within the KDC. The client constructs a TGS-REQ (Ticket Granting Service Request) message, including the TGT received earlier and a request for a Service Ticket for the desired service as seen in the Figure 9.

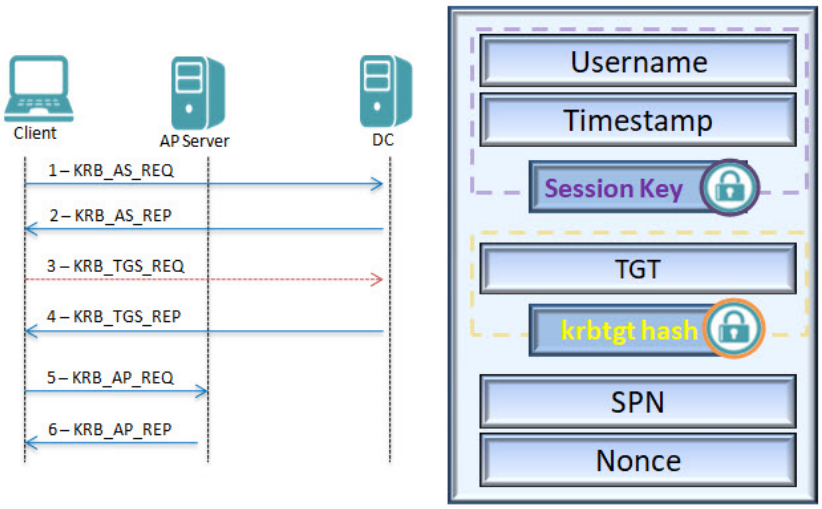

Figure 9: Kerberos TGS Request

4. **Ticket Granting Service (TGS) Response:** The TGS within the KDC verifies the TGT and the user's request. If everything is valid, it creates a Service Ticket and sends it back to the client. The KDC sends a TGS-REP message which can be seen in Figure 10 that containing the Service Ticket encrypted with a session key derived from the user's TGT.

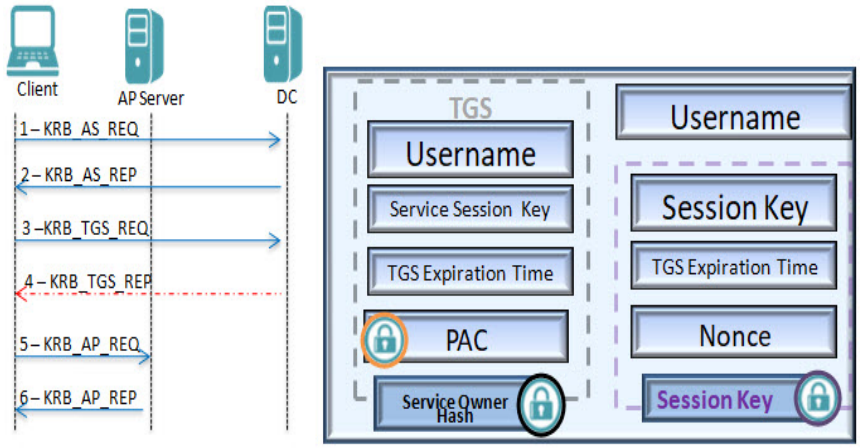

Figure 10: Kerberos TGS Response

5. **Accessing the Service:** With the Service Ticket in hand, the client sends a request to the target service as seen in the Figure 11, presenting the Service Ticket as proof of authentication along with an Authenticator. The service validates the Service Ticket and the Authenticator. If both are valid and the user is authorized to access the service, the service grants access.

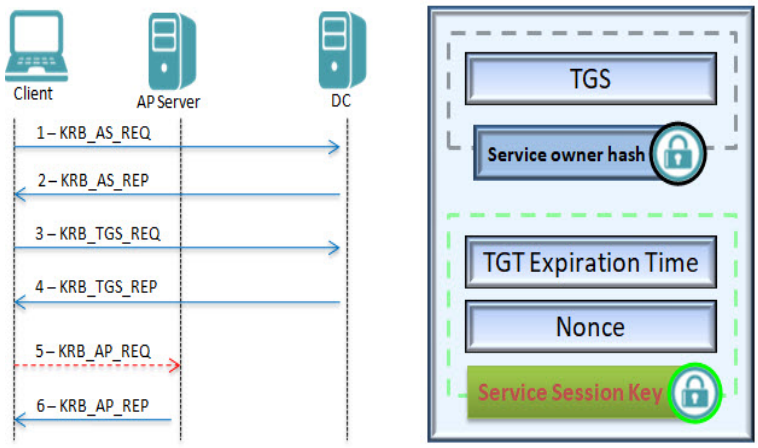

Figure 11: Kerberos Application Request

6. **Session Establishment:** Once the service validates the Service Ticket and Authenticator, a secure session is established between the client and the service using the session key embedded in the Service Ticket. This session key is used to encrypt further communication between them.

We can also explain the question of how to authenticate a user with the Kerberos protocol with a simple analogy as follows; Imagine Alice, a diligent student at university who is keen to learn about many subjects. Bob, the institution's human resources administrator who is renowned for allowing entrance to the university, is asked for help by Alice. Bob gave Alice a unique "Master Pass" called the Academic Passport (also known as the Ticket Granting Ticket, or TGT). With the university's symbol on this Academic Passport, Alice can ask to attend different classrooms without requiring additional approvals for each. Bob the attentive administrator checks Alice's Academic Passport and, after making sure everything was correct, give out individual passes for every subject area. These passes have emblems on them that stands for every academic discipline. By obtaining the Academic Passport (TGT) and Academic Service (Service Tickets-TGS) tickets, Alice has the opportunity to attend different classes representing different subjects. At the door of each classroom, faculty members verify the authenticity of her pass cards, allowing her to attend classes related to her academic field.

2.3.4 Common Vulnerabilities and Exposures about Kerboros Protocol

The concepts of Cryptographic perimeter, Active Directory and Authentication with Kerberos have been tried to be explained so far. Before moving on to the Kerberoasting attack, I would like to briefly mention the vulnerabilities that have recently occurred with the Kerberos protocol, so that the subject can be understood better. Here are a few Common Vulnerabilities and Exposures (CVEs) related to the Kerberos protocol [51]:

CVE-2020-17049: This vulnerability affected the Kerberos KDC in Microsoft Windows and was caused by a flaw in how KDC handles AES encryption types. Exploiting this flaw could allow an attacker to bypass Kerberos authentication.

CVE-2014-5352: This vulnerability involved a security bypass issue in the Kerberos 5 Key Distribution Center daemon. It allowed remote attackers to impersonate users or services by leveraging the use of multiple realms in a single KDC.

CVE-2014-4341: This vulnerability impacted the implementation of the KDC in MIT Kerberos 5 before 1.12.5. It allowed remote attackers to cause a denial of service (KDC daemon crash) via a crafted AS-REQ request.

CVE-2012-1013: This vulnerability was found in the Kerberos KDC in MIT Kerberos 5 before 1.10.4. It allowed remote attackers to cause a denial of service (NULL pointer dereference and daemon crash) via a crafted request.

It is crucial keeping systems updated with security patches and following best security practices is crucial to mitigate the risks associated with CVEs and vulnerabilities in any protocol or software, including Kerberos.

CHAPTER 3

KERBEROASTING AND DETECTION METHOLOGIES

Kerberoasting is a critical cybersecurity threat that exploits weaknesses in the Kerberos authentication protocol within Active Directory environments. So, understanding Kerberoasting authentication protocol has importance for defense. This overview focuses on explaining the Kerberoasting attack, discussing detection methods to identify vulnerabilities, and exploring the tools and technologies frequently employed by attackers in executing this threat.

3.1 Understanding Kerberoasting Attack

Tim Medin, initially presented the Kerberoasting attack in 2014 [52]. Since researcher Tim Medin first revealed the approach at the 2014 DerbyCon conference in Kentucky, advanced threat actors, penetration testers, and red teams have all adopted it as a common tool in their toolbox. Kerberoasting allows privilege escalation and lateral network movement, just as the majority of attacks against AD [53]. This attack is still valid and often works well even after many years have pass since AD environment administration is often underestimated. Its objective is to break passwords for remote service accounts entirely offline, without requiring special or elevated rights, and without submitting a single packet to the service. Because any user on a domain may use the method and it does not require the privileges of an administrator, it is widely used and makes it easy for malicious actors with any degree of expertise to take advantage of [54].

As explained in Chapter 2, the domain concept was created to access resources offered on the network. Thus, with the concept of domain, a secure border is provided for access to these resources. Kerberos protocol is used by default in AD domain which is the default authentication method in AD and it offers smooth SSO (Single Sign On) access to services through network logon. Kerberos authentication is explained in detaily in section 2.3.3. As can be understood from there, any user who authenticates to the domain with a valid TGT ticket will want to access the services offered in the domain such as mail, web, dns, imap, pop, etc. In order to access such services which are identified by Service Principal Name (SPN), client will request from KDC service the TGS ticket of the service to access with the TGT ticket it has. Windows uses SPNs to determine which service accounts are used to encrypt TGS tickets. SPNs can be connected to host-based accounts or domain user accounts. However, normal user accounts are generally used for service accounts in AD. Because many applications are not compatible with working with a host-based service account. An attacker with access to the compromised account can request a TGS service ticket, which usually works with a normal user account. Because generally it is not possible to crack the password of services that operate with a host-based account, consisting

of 128 characters, within valid 30 days. The service ticket which is operates with domain user account could be encrypted using a weak cipher suite, like RC4-HMAC-MD5, in which case the NT password hash of the service account is used to encrypt. There are 5 cipher suites used in AD are listed in Table 1. These are RC4, DES and AES. Of these symmetric encryption algorithms, only AES is secure. DES and RC4 ciphers are no longer used. However, windows has a backwards compatibility policy setting that allows these packages. Because there are technologically obsolete applications that are in use. And also these applications may apply obsolete encryption algorithms in companies and institutions. Briefly, Kerberoasting takes advantage of the fact that these service account passwords are often weak or easily guessable. Once these encrypted tickets are obtained, the attacker can attempt to crack the passwords offline using brute-force or dictionary attacks. Successful decryption of these tickets reveals the plaintext passwords, granting unauthorized access to the targeted service accounts.

In the Kerberoasting attack, adversaries take advantage of original Kerberos protocol, which mostly depended on symmetric key cryptography, rather than any specific vulnerability. To address them, PKINIT (Public Key Cryptography for Initial Authentication in Kerberos) was developed [55]. By utilizing public key cryptography's advantages, PKINIT tackles this problem. Through PKINIT, a client uses its private key to sign a timestamp as verification of identification for the Key Distribution Center. With a copy of the client's public key, the KDC can authenticate the client by confirming the signature. This removes the requirement for a beforehand symmetric key sharing between the client and KDC. For PKINIT to work in an AD environment, domain controllers must be configured with the appropriate public key infrastructure (PKI) and certificates. Many AD infrastructures lack this configuration and therefore do not utilize this security mechanism. So unsufficient AD management within enterprises is the primary cause of Kerberoasting's success. Beyond this numerous service account credentials being the same length as the minimum domain password andd service accounts frequently lack password expiration settings, which is another issue. In addition, most service accounts have excessively extensive permissions; they have access to certain objects or privileges comparable to those of an administrator [56]. Another advantage in terms of attacker is, to compromise an AD domain, an attacker just has to take over a legitimate regular account or know the login credentials for one. Attacker can submit one or more service ticket (SPN) requests using this account. Briefly, Kerberoasting a common attack method that targets credentials for AD service accounts. Because anybody on a domain can do Kerberoasting, not only administrators, both more experienced and less experienced attackers prefer it. Additionally, as it is a "offline" attack, no packets need to be delivered to the targeted service, traffic that may potentially cause alarms to be logged. Alternatively, Kerberoasting leverages known security flaws in Kerberos authentication for AD almost as much as it does human nature. Fundamentally, a password-cracking exploit known as "Kerberoasting" involves stealing credentials from memory and cracking them offline. The practical operation of a Kerberoasting attack is as follows in Figure 12 which is adopted from [11]:

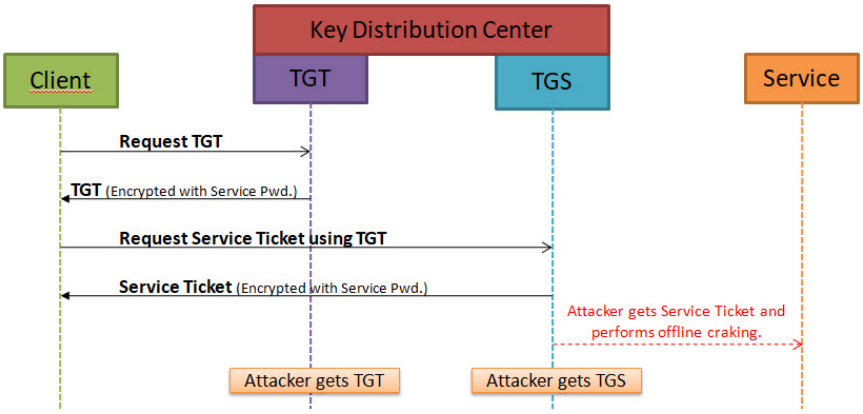

Figure 12: Kerberoasting Attack Diagram

1. An attacker first gains access to a domain user's account. It is not necessary for the user to have elevated or "administrator" rights. The attacker obtains domain authentication.

2. The malicious user receives a TGT from the KDC signed by its KRBTGT service account in AD once they have successfully authenticated by compromised user account.

3. In order to breach a service, the attacker seeks a service ticket especially with the weak encryption algorithm. In order to generate a TGS ticket and encrypt it using the service password, the domain controller will get the permissions from the AD database. Since only the domain controller and the service have the secret, only they are able to decrypt the ticket. Vulnerable user accounts in the AD domain can be found in a few different methods. Using a program like BloodHound, which can map out the ties of trust between various domains and pinpoint high-value targets, is one approach. A different approach is to utilize a program such as PowerView, which can look for user accounts with SPNs and retrieve details about the services they are connected to [57].

4. The user receives the service ticket from the domain controller, which is then forwarded to the service, which decrypts it and decides if the user is authorized to use the service. At this stage, the ticket may be taken out of system memory by an attacker, who would then crack it offline. An attacker can use a program like Rubeus or Kekeo to open a service ticket for a user account after they have located one that is susceptible. The attack can be made more effective by using these tools to submit service ticket requests for several user accounts at once.

5. Tools that automate the process of password cracking include Impacket, PowerSploit, and Empire. These tools ask for service tickets and return hashes of crackable tickets in formats that can be submitted to tools like Hashcat and John the Ripper, which will extract plaintext credentials from hashes that are vulnerable.

To briefly summarize, an attacker with access to a valid Kerberos ticket, even a low privileged user, can request a TGS ticket for any service in a Kerberoasting attack. Since the password of the service account encrypts the TGS ticket, an attacker can take the ticket and try to crack it offline without worrying about being discovered. If the service account is weak, the attacker can gain unauthorized access to the service by cracking its password [11].

3.2 Detection Techniques

The two main kinds of intrusion detection techniques now in use are anomaly detection and signature recognition [58]. Anomaly-based detection looks for behavioral changes, whereas signature-based recognition looks for risks that we are aware of. A preprogrammed set of recognized indications of compromise is the foundation of signature-based recognition. An indication of compromise may consist of malicious network attack activity, email subject line content, file hashes, malicious domains, or known byte sequences. Alerts about network traffic, such as known malicious IP addresses trying to access a system, may also be included in signatures [59]. In order to use signature recognition algorithms, recorded attack signatures are compared to observed events. When there is a match, the strategies alert users to an incursion. Obvious disadvantages of these methods is their inability to identify novel attacks with unidentified signatures [60]. In contrast, anomaly detection creates models using normal data and looks for any deviations in the observed data from the normal model. The objective is to ascertain if a new piece of test data belongs to "normal" behavior or to an anomalous behavior given a collection of normal data to train from. Anomaly-based detection involves first training the system with a normalized baseline and then comparing activity against that baseline. Once an event appears out of the ordinary an alert is triggered. Alerts can be triggered by anything that does not align with the normalized baseline. The benefit of anomaly detection approaches is their ability to identify novel forms of incursions as variations from typical usage [61]. Their vulnerability, though, is the high occurrence of false alarms.

3.2.1 Signature-Based Recognition

In the context of a Kerberoasting attack, signature-based recognition refers to a technique employed by security systems to recognize, prevent, or issue an alert against particular patterns or signatures connected to this kind of cyber threat. These signatures may take the form of particular activity sequences that are suggestive of an attack, network traffic patterns, or anomalies in authentication logs. These signatures can be recognized by security systems, such as intrusion detection and prevention systems (IDS/IPS) or security information and event (SIEM) platforms, which can then be set up to raise alarms or take countermeasures to the attack. Signature recognition rules contain specific conditions that configured based on windows logs collected from machines over the network. In the study [14], a set of signature-based rules were developed that can be used to detect Kerberoasting attack. I would like to briefly summarize this study as follows;

3.2.1.1 Detecting via TGS Requests with Weaker Encryption Algorithm

Since all Kerberos authentication requires the use of AES cipher suites, as was previously indicated, any requests for TGS tickets that employ a lower encryption type should be viewed with suspicion. The windows event log (Event ID 4769) should raise red flags if the ticket encryption algorithm request is of the type 0x1 (DES_CBC_CRC), 0x3 (DES_CBC_MD5), or 0x17 (RC4_HMAC_MD5), as indicated in Table 1. This means that the encryption technique requested is not AES as seen in the Figure 13. First and foremost, of course, the whitelist should include services that utilize outdated encryption techniques in the environment, and any requests made to such services should be closely monitored.

Figure 13: Weaker Encryption Algorithm in Kerberos Protocol

3.2.1.2 Requests for Suspicious Service Tickets

This detection approach is based on two suspicious behaviours. One source's excessive service ticket requests are the first, while suspicious requests for service tickets from outside sources are the second.

The first behavioral pattern is initiated when a single resource generates a large number of different service requests in a short period of time. Such behavior is even more suspicious if the service names have no relation to each other or if the requested services are not specific to that provider. Because a client requesting multiple and unrelated services in a short time should bring to mind the idea that the client is exploring the services in the domain.

The second one, which relates to suspicious external service ticket requests, is in accordance with a security guideline that Microsoft has detailed in its Event ID 4769[1] literature [62]. The network data given during the incident is the main focus of the search. It keeps an eye on the use of popular ports and records any instances in which the IP address is outside of the private IP ranges indicators of an outbound connection.

3.2.1.3 Using a Honeypot to Detect Kerberoasting

Sean Metcalf provides a useful technique for identifying Kerberoasting in one of his studies [63]. He recommends setting up a fake account called a "honeypot," which would be linked to a fictitious SPN and have certain properties (like "AdminCount[2]") configured to entice potential attackers. This account was made just to draw in attackers; it serves no useful purpose in the ecosystem and has no rights. When service tickets are requested for this account, there is no justifiable reason to seek them, so tracking these requests yields unambiguous evidence of malicious activity and a low false positive rate.

3.2.1.4 Utilizing PowerShell Logs to Identify Kerberoasting

This detection technique aims to capture SPN scanning activity or successful acquisition of the service ticket hash by utilizing PowerShell logging features. PowerShell events 4103 and 4104 [3] are located, and a full-text search is conducted inside them to find strings that include service account names. From a single workstation, transactions are generated for every PowerShell event that occurs later. If more events exceeding the predetermined threshold have matching strings, results are generated. As an input, a list of service accounts and SPNs has to be prepared.

Information about signature-based attack recognition methods for detecting Kerberoasting attacks was given in the previous section. By changing the attack signature, attackers might slightly vary their techniques in order to avoid detection. For this reason, depending only on signature-based recognition techniques could not be sufficient to defend against more complex or developing threats. The other restriction is, rules relying on signatures may have a high false alarm rate. The rules need to be able to detect malicious activity even when attackers utilize authentic machines and user accounts. However, these rules may also set off recurring events in these accounts, such as incorrect password login problems or administrator-initiated actions. A real alert may be overlooked or disregarded if there are too many false alarms. So, in order to effectively prevent cyber threats such as Kerberoasting, it is crucial to enhance security methods such as signature-based recognition with additional security measures such behavioral analysis, anomaly detection, frequent security updates, and employee training.

[1] Event ID 4769 in Windows Security logs corresponds to a Kerberos service ticket request. Specifically, it indicates that a Kerberos service ticket was requested and successfully granted. This event is part of the Windows security auditing system and is logged when a user or computer requests a service ticket to access a service.

[2] In Active Directory, AdminCount=1 signifies that an object is a member of a privileged group, triggering the AdminSDHolder process to enforce predefined security settings on that object to protect it from unintentional modifications by ensuring consistent security descriptors. Objects with AdminCount=1 are typically associated with critical administrative roles within the domain.

[3] Event ID 4104 will include the entire script, while Event ID 4103 will show details of the pipeline execution, including command invocations and variable initialization, as PowerShell runs.

3.2.2 Anomaly Based Detection

Anomaly detection is a technique used in various fields, including cybersecurity, finance, healthcare, and more, to identify patterns, events, or observations that deviate significantly from the norm or expected behavior within a dataset. In the context of cybersecurity, anomaly detection aims to pinpoint activities or events that differ from regular patterns and could indicate potential security threats or abnormal behavior. Approaches to anomaly detection:

3.2.2.1 Statistical Methods

Network anomaly detection using statistical methods involves identifying unusual or suspicious patterns in network traffic data. The goal is to detect deviations from normal behavior that may indicate potential security threats or abnormal activities. An activity profile describes the typical behavior of a monitored parameter, which helps to understand what that parameter's subject typically does. A statistical measure and model can be used to characterize observed behavior. The time between two audit-related occurrences or a set interval of time (seconds, minutes, hours, etc.) can be used as the observation period [64]. For example, between opening and closing a file, between logging in and out, etc. Several statistical techniques are employed for network anomaly detection: Traffic Baseline Analysis, Statistical Thresholds, Flow-based Analysis, Protocol Analysis, Entropy-Based Methods, etc.

3.2.2.2 Machine Learning Methods

Supervised Learning: Uses labeled data to identify anomalies. A labeled training set of both normal and anomalous samples was needed for supervised methods, commonly referred to as classification methods, in order to build the predictive model. Because they have to access more information, supervised methods should theoretically have a higher detection rate than semi-supervised and unsupervised methods. But there are certain technical problems [65] that make these approaches appear to be less precise than they should be. Lack of a comprehensive training data set is the first problem. Furthermore, it might be difficult to assign appropriate labels, and disturbances in training sets frequently lead to greater false alarm rates. The most widely used supervised algorithms are decision trees, k-Nearest Neighbors, Bayesian networks, Supervised Neural Networks, and Support Vector Machines [66]. This is less common in anomaly detection as anomalies are usually rare and difficult to label.

Unsupervised Learning: Data for training is not required for these methods. Alternatively, they were predicated on two fundamental principles. Initially, they assume that a relatively small number of network connections are irregular and that the majority of traffic is normal. Secondly, they expect malicious traffic to differ statistically from benign traffic. These two presumptions state that data groups of like instances that occur regularly are thought to be normal traffic, whereas rare cases that differ significantly from the bulk of instances are thought to be malicious [67]. Utilizes algorithms such as K-Means Clustering, Self-organizing Maps, C-means, Expectation-Maximization Meta algorithm, Adaptive Resonance Theory, Unsupervised Niche Clustering and One-Class Support Vector Machine to identify outliers without prior training on labeled data [68].

Semi-Supervised Learning: The area of machine learning known as semi-supervised learning works with both labelled and unlabelled data to carry out certain learning tasks. From a conceptual standpoint, it lies in the middle of supervised and unsupervised learning, allowing for the utilization of both normally smaller quantities of labelled data and the vast volumes of unlabelled data available in many use cases [69]. It incorporates a small amount of labeled data combined with a larger amount of unlabeled data for training. Semi-Supervised ML provides an advantage when labeled data for Kerberoasting attacks is limited. The model can learn normal behavior without explicit attack labels and identify anomalies based on deviations from the learned patterns. Fine-tuning the model and ensuring continuous learning are key aspects of an effective Kerberoasting attack detection system using Semi-Supervised Learning.

3.2.2.3 Classifier-Based Anomaly Detection

The concept that typical characteristics behavior can be separated from abnormal behavior is the foundation of anomaly detection. A classifier can be used to forecast the typical forthcoming event. Anomaly occurs when an occurrence during the monitoring phase differs from expected outcome of classifier. The following steps are usually involved in the classification process [60]:

1. Using training data, identify classes and class properties.
2. Determine the characteristics to be categorized.
3. Make use of the training data to build a model.
4. Classify the unknown data samples using the learned model.

3.2.2.4 Identifying Anomalies With Finite State Machines

A model of behavior consisting of states, transitions, and actions is called a finite state machine (FSM). In this framework, a state contains historical data, and a transition denotes a change in state and is accompanied by a condition that must be met in order for the transition to occur. An action is a description of a task that needs to be completed at a specific time [60].

FSM can be utilized for Kerberoasting detection by defining states representing various stages in the Kerberos authentication process. The FSM models normal behavior and introduces additional states or transitions to detect anomalies indicative of Kerberoasting attacks, such as an unusual number of service ticket requests or extended durations between requests. Monitoring the sequence of states in real-time allows for the detection of deviations from expected behavior, triggering alerts or responses when potential Kerberoasting activity is identified. Fine-tuning parameters, integrating with security infrastructure, and establishing a feedback loop for continuous improvement enhance the effectiveness of Kerberoasting detection using FSM.

CHAPTER 4

EXPERIMENTAL STUDY

In this section, a study is going to be carried out to detect the Kerberoasting attack with machine learning algorithms. For this purpose, following a brief explanation of the method used in the experimental study, the process of creating the dataset, feature engineering and appropriate feature selection, hyperparameter tunning, explanation and implementation of machine learning models and their results are going to be explained. The aim of the thesis is to determine the most efficient ML algorithm for detecting Kerberoasting attack using with supervised ML techniques. Studies have been carried out in this area, as mentioned in the previous sections. However, since the previous study was conducted in a corporate environment, the dataset of the Kerberoasting attack could not be shared. There is no publicly available dataset that can be used for the Kerberoasting attack to measure the efficiency of any ML algorithm. For this reason, the thesis study was carried out in a virtual environment to create a dataset. Thus, datasets containing security logs were created in the Azure cloud environment and made publicly available. [1]. Due to the size of the dataset, just Event ID 4769, which was associated with the Kerberoasting attack, was disclosed due to the size of the security log collection.

A brief explanation of the road map followed for the creation of the laboratory environment will be useful for easier understanding of the work carried out in this section. A virtual environment consisting of four machines was created in Azure; one domain controller that represents DCs in a domain, one application server to represents the servers that runs in the domain, one client machine to represent the users accessing the services in the domain and a Kali machine to represent the attackers who exploit vulnerabilities in the created domain. Eight service accounts have been created on the service server to represent the services that users can access in the standard information system environment. Three of the service accounts are created to represent service accounts that work with the AES encryption algorithm by default. However, to represent the vulnerabilities in the service accounts in the IT environment that had to be used due to backward compability, three services were selected as service accounts that work with weak encryption algorithms such as DES and RC4. In the following stage, 74 weak encryption algorithm TGS requests were made on behalf of the compromised user during 21 days for access to services working with weak encryption algorithm, through the account of a user whose password was obtained from attacker via Kali machine by various methods. Later, TGSs were cracked offline due to the RC4 weak encryption algorithm obtained seen in Figure 14.

[1] Dataset available at: https://github.com/ysnakst/Dataset-for-Kerberoasting/blob/main

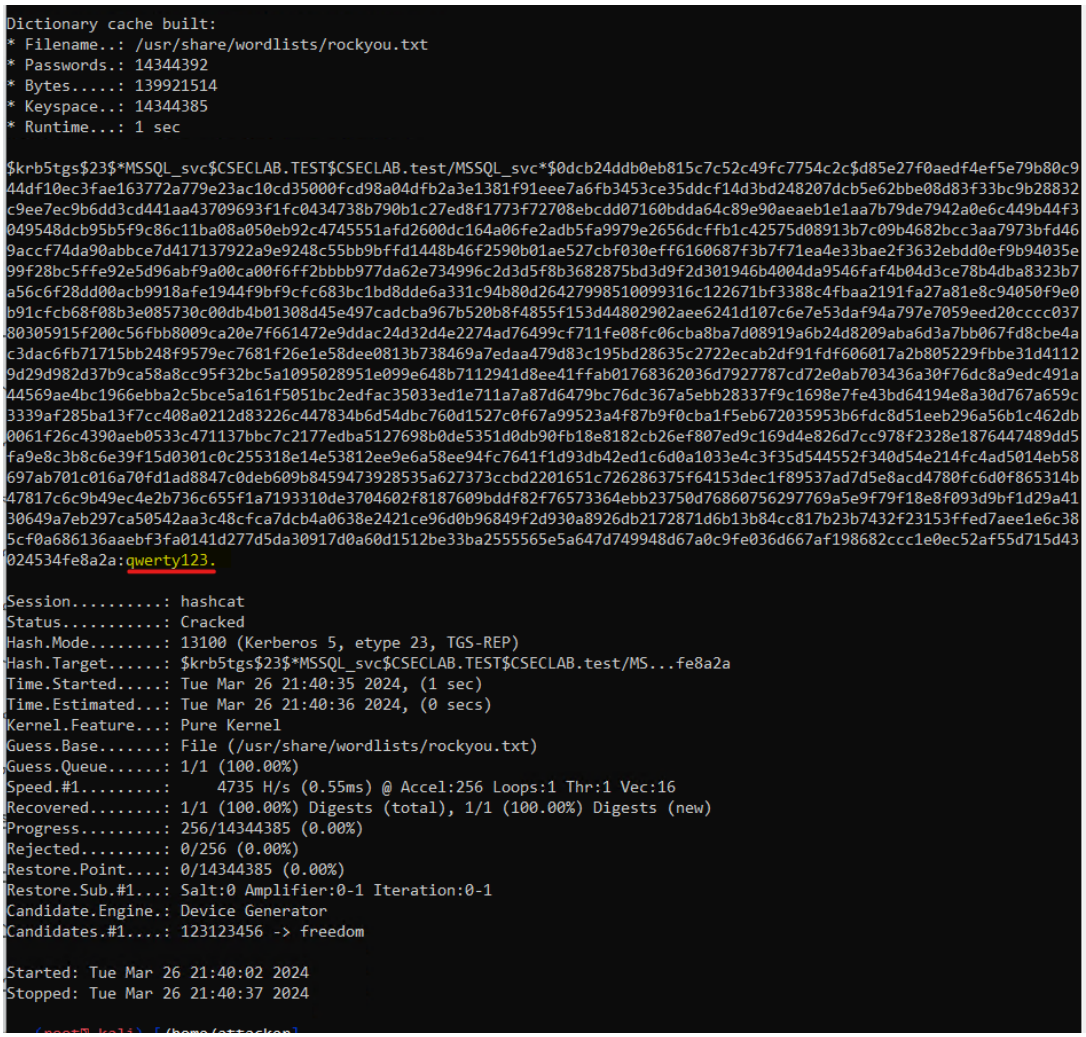

Figure 14: Offline Password Cracking

Security logs created in all these processes were transferred to the ELK Stack environment, which were also deployed in the cloud environment. Security logs obtained from the servers for 21 days and exported in order to utilize as a dataset in machine learning algorithms. The obtained data was transformed into meaningful data for the detection of Kerberoasting attack through Jupyter Notebook, using the method called feature engineering. And then, the security log dataset was transferred to the SIEM environment to be used in ML algorithms with Scikit-learn ML python library. The efficiency of ML algorithms was compared using the statistical data obtained as true positive, false positive, true negative, false negative values in the prediction data.

To start working in order to achieve significant and efficient results to achieve the goal, it will be necessary to resort to an accepted scientific method. In addition to being an open standard process model expressing common approaches in data mining, the Cross Industry Standard Process for Data Mining (CRISP-DM) model, the most used analytical model, will be used. The CRISP-DM is a process model that serves as the base for a data science process. As seen in Figure 15 which is adapted from [70] has six sequential phases :

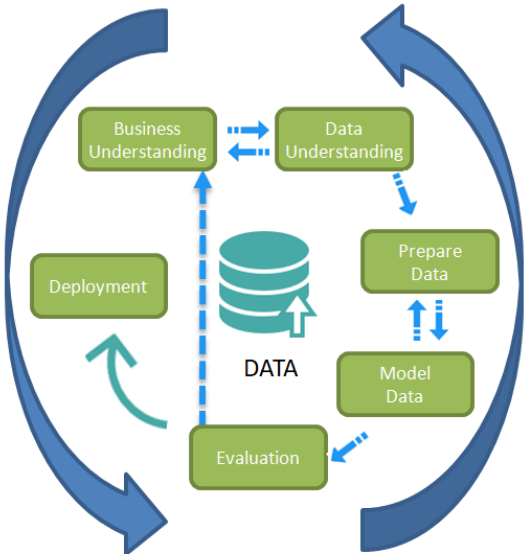

Figure 15: CRISP-DM Methodology

1. Business understanding – What is needed by the organization?
2. Data understanding – What data is needed or do we have? Is everything clean?
3. Data preparation – How should the data be arranged for modeling?
4. Modeling – Which modeling techniques ought to be used?
5. Evaluation – Which of the models best satisfies the organization goals?
6. Deployment – How are the outcomes made available to stakeholders?

4.1 Business Understanding

In order to define the problem and develop a business plan for the data mining process, this step entails comprehending the objectives and requirements of the project. It is critical to consider the aims in terms of problem solving, comprehend the study's intended audience, and reduce this issue to a data mining issue. While working on them, we have to ascertain what data is available and what potential issues might come up throughout the project.

The aim of this thesis is to determine the most efficient algorithm by comparing the detection methods of Kerberoasting attack with supervised MLs. In the study, the efficiency of supervised ML algorithms are going to be compared with each other. Considering that each environment will have its own characteristics, it would be more appropriate to use supervised ML algortithms. In this way, a guide going to be provided for the IT is environment to create the most appropriate supervised ML for itself. In order to detect an attack, the ML algorithm needs to learn the system normals and abnormalities specific to each environment in a short time and generate an alert to a cyber security analyst when an abnormal situation occurs. In the study, semi-supervised and unsupervised algorithms were not preferred due to

the need for data and time to learn the normalities of the information system environment. Because, even a missed attack can cause a full domain compromise. For this reason, it was thought that it would be more correct to prefer supervised algorithms. Supervised ML yields significantly more precise and credible results than unsupervised learning since it makes use of known and labeled input data [71]. Some studies have been conducted to compare the performances of supervised, semi-supervised and unsupervised algorithms with each other. In order to decrease false alarm ratio further, we tried other supervised machine learning algorithms like *Logistic Regression, Gaussian Naive Bayes, Bernoulli Naive Bayes, Support Vector Classifier (SVC), Random Forest Classifier (RFC), Extreme Gradient Boosting Machine (XGBoost), K-Nearst Neighbours (KNN), Light Gradient Boosting Machine (Light GBM)* and *Decision Tree Classifier* to analyze their efficiency in terms of detect Kerberoasting attacks.

The preceding sections provided a detailed explanation of the problem. The generation and appropriate modification of the dataset presents largest challenge to reaching the aim en route to the goal. During the data preparation and modeling phases, a wide range of technologies that are often employed in machine learning have been used. Python, which is widely used in data science, is used in the machine learning technology stack together with *Jupyter Notebook, Scikit-learn*, and other libraries like *Pandas, NumPy,* and *Matplotlib*. These technologies made it possible to automate procedures, improve feature engineering, conduct more extensive testing, and thoroughly analyze various strategies.

4.2 Data Understanding

As stated in previous sections, it was explained that the main basis of the attack was that system administrators prefer normal user accounts using with a weak encryption algorithm instead of machine or host-based accounts to run their services, for reasons such as backward compatibility. In the study, an experiment will carried out on the efficiency of machine learning algorithms in terms of detecting Kerberoasting attacks. Therefore, there is a need to create a laboratory environment to simulate the attack. Based on this idea, a virtual environment shown in Figure 16 has been created in Azure Cloud, including domain controller, server, attack machine and client. Cloud-based ELK (Elastic Search-Logstash-Kibana) Stack servers were used as a SIEM environment to collect the log records to be obtained during the experiment.

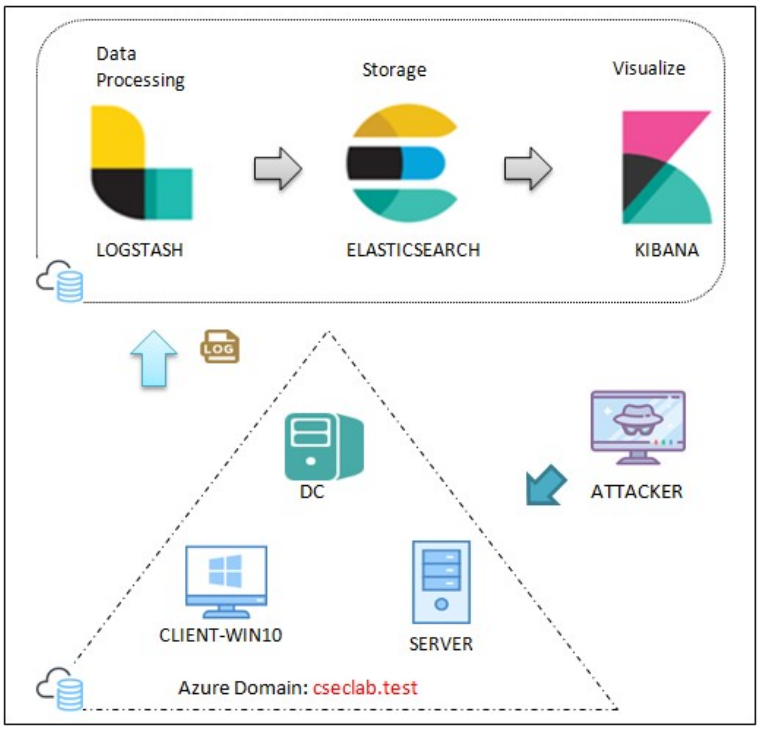

Figure 16: Virtual Lab Environment

In the virtual environment, user account-defined service SPNs were assigned, including DHCP, File, Exchange, IIS, VMware, MSSQL, Printer and Backup services which can be used in even a small scale IT environment where an attacker can detect and exploit the vulnerability. In Table 4, the common services are listed that will might be used by standard users and also some the services that served as honeypot services are listed to detect the attack. In order to represent backward compatiable systems

Table 4: Services in Lab Environment

Services	**Cipher Suite**
DHCP	Strong
FILE SERVER	Strong
EXCHANGE	Strong
IIS	Weak
VMWARE	Weak
MSSQL	Weak
PRINTER	Weak
BACKUP	Weak

that can be used in every ordinary IT environment; DHCP, File, Exchange services were randomly determined as the normal services and remaining services are selected as honeypot services to detect any attack in the experiment. Event ID 4769 records for all service requests were collected for 21 days.

4.3 Data Preparation

Because data preparation directly impacts the model's accuracy and performance, it is crucial for machine learning. The data preparation process consists of 5 stages as seen in Figure 17.

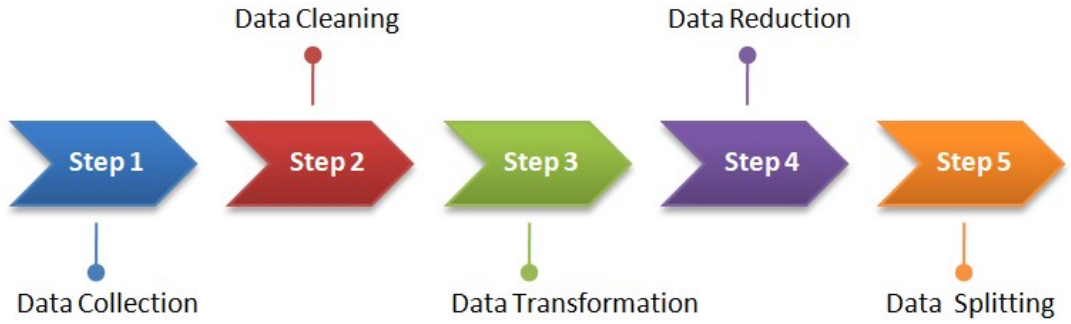

Figure 17: Data Preparation Process

4.3.1 Data Collection

In the IT environment that is deployed on the cloud platform, all Event ID 4769 records "Kerberos service ticket was requested" which are the only indicator by which the Kerberosting attack, were audited from domain controllers. Ticket requests of all *krbtgt*, machine accounts and user accounts were transferred to the ELK Stack platform via *WinlogBeat* [2]. Although it is a very small laboratory IT environment, only Event ID 4769 reached 5994 rows. Since the Kerberoasting attack only targets tickets with weak encryption algorithms, filtering *DES* (0x1, 0x3), *RC4* (0x17) encryption algorithms seems to be a appropriate approach. There is a point to note at this phase; A TGS ticket with the AES algorithm should not mean that it cannot be decrypted. The AES algorithm can also be cracked, but it will take quite long time. However, due to the possibility of high data imbalance in the datasets negatively affecting the ML training phase [72], all service requests with *AES* (0x11, 0x12) encryption algorithm were also included. Thus, it would be appropriate to transfer the security events of the Kerberos ticket request to be used in the supervised ML model in a natural structure. Following the selection of the events, feature engineering for machine learning and the extraction of data required for Kerberoasting detection must be done. To effectively filter and extract features for ML models, it is crucial to comprehend the format and potential values that can be included in events. As seen in Figure 18 there are some details regarding Event ID 4769.

[2] Winlogbeat is a lightweight data shipper within the Elastic Stack, tailored for collecting Windows event logs. It efficiently gathers various types of event logs, including system, security and application logs from Windows machines

Figure 18: Event ID 4769 Kerberos Service Ticket Request

The user who made the service request is shown with the domain name and UPN in the *Account Name* field. In the example above, the user john.smith is a user that is defined in the cseclab.test domain. Considering this details, some of vaulable data that are related to user who made the service request can be obtained. As can be seen, the *Service Name* field contains the name of the requested service. In the example, user John Smith try to reach the file server service. As seen in the *Client Address* field, the IP address is specified. In the additional information section, there are also *Ticket Encryption Type* and *Ticket Options* fields. In the Figure 18, the field marked as *0x17* indicates that TGS is encrypted with the RC4 algorithm [73]. Attacker needs to make service requests much more times to find the TGS tickets that are encrypted with a weak encryption algorithm. Additionally, the *Ticket Options* field gives different outputs depending on the request method. Taking this into account, if anyone who wants to get an idea regarding which channel the attacker utilized during the service request can review this field. Considering all these points, a static query rule was applied as shown in Figure 19 to obtain all Event ID 4769 records from the DC's;

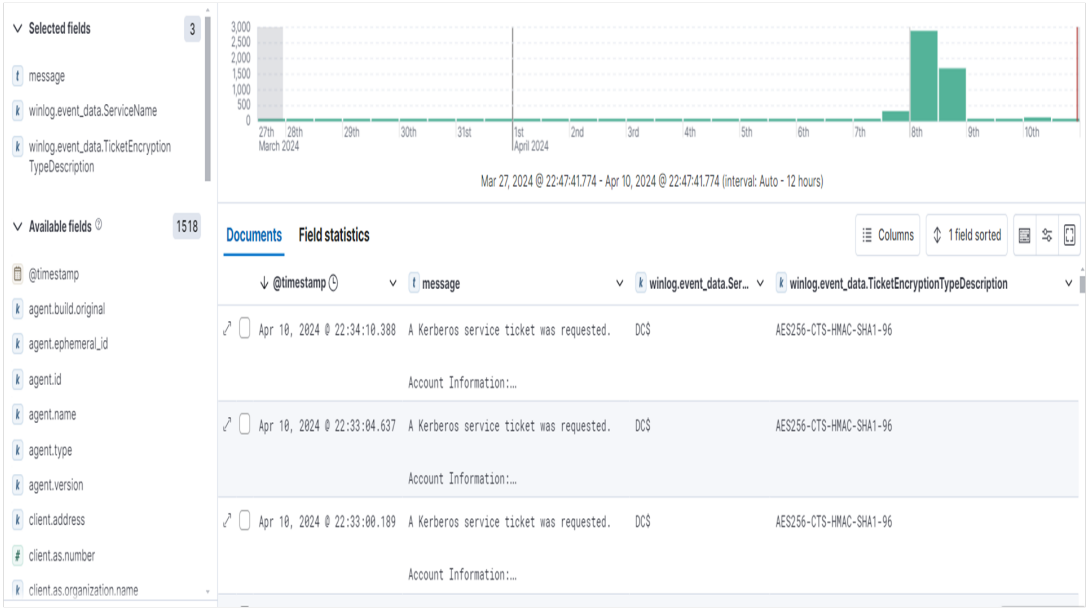

Figure 19: Service Ticket Requests of Domain

As seen Figure 19 there are 5994 records of Kerberos service ticket request for 21 days just for three machine. If a large IT environment with 50k or more users is considered, the volume of event logs would reach quite large sizes.

4.3.2 Cleaning Data

After collecting the data, the next step is going to be to clean the data and convert it to an usefull inputs. In this part, if it contains any missing values, outliers and inconsistencies in the dataset will be identified and addressed. To begin with *computer accounts* must be deleted from the collected log records. Because such account's passwords are automatically changed by the server operating system with strong password in every 30 days and with a new password that has a maximum length of 256 characters [74] and using default AES encryption algorithm. Since the attacker will not be able to obtain the password in such a short time, it would not be logical for the attacker to choose computer accounts when these weak encryption algorithms are available. Accounts with a dollar ($) sign at the end of their names have been filtered out from our dataset because they represent computer accounts.

When the log records regarding the Kerberos ticket request event records are examined, it appears that the "krbgt" account, which looks like a user account due to the lack of a dollar sign at the end, has also been recorded many times. As mentioned in the previous sections, the krbtgt account is the account that runs the kerberos service. It uses the AES encryption algorithm unless manual configuration done. Since it is the most crucial account of the domain, it is monitored in detail by many cyber security monitoring teams. For this reason, it would be an appropriate course of action to filter the log records of the krbtgt account.

After determining data cleaning criteria, datas are left with only Event ID 4769 records for service accounts. The KQL query that contains requests for Kerberos service tickets using DES and RC4 encryption algorithms in the laboratory environment in the study is shown in Figure 20.

Figure 20: Service Ticket Requests with Weak Encryption Algorithm

4.3.3 Data Transformation

Cleaning data and transforming it into a format that machine learning algorithms can utilize it as a meaningful data is known as data transformation. The performance of the ML model can be directly impacted by the preparation of the data, which is why data transformation is an important step. The improper outcome will stem from incorrect data conversion. It requires some efforts to transform data without altering the parameters that determine the intended outcome. Kerberos ticket requests for services cannot be utilized simultaneously by the same client that use a weak encryption technique. This is the most noticeable symptom of the Kerberoasting attack, and it happens in a very short time period. Taking these concerns into account, data transformation is going to be handled in 2 stages as stated in the following headlines.

4.3.3.1 Feature Engineering

During the feature engineering phase, features that can identify the normal and abnormal behavior of the recorded activity are extracted in an effort to increase the probability of the machine learning algorithm that will yield effective results. Weak encryption types, the volume of service tickets that requested in short time period, the services involved, and the user accounts requesting them are the most obvious clues to look out for when identifying Kerberoasting attack. Because Event ID 4769 records are the only indication of a Kerberoasting attack. The fields of the Event ID 4769 records that we can be used for detecting Kerberoasting attacks are listed in Table 5.

Table 5: Feature Engineering

Feature	**Description**
acc_system	Is account system account?
acc_personal	Is account real user account?
acc_computer	Is account service account?
svc_app	Application services like dhcp, iis etc.
svc_kerb	Kerberos authentication service
ticket_secure	Encyption of Kerberos ticket is reliable?
ticket_unsecure	Encyption of Kerberos ticket is unreliable?
svc_req_count	Service request count
dist_svc_count	Distinct count of service request
svc_count_ratio	Ratio of dist_svc_count and svc_req_count
duration	Time span of experiment
last_time_dif	Total time difference between the last service ticket requests
total_time_dif	Cumulative total time difference between the last service ticket requests
time_max	Maximum time difference between the last service ticket requests
time_min	Minimal time difference between the last service ticket requests
time_avg	Average time difference between the last service ticket requests
issuspicious	Unusual actions or behaviors that could indicate a potential security breach

The account type of the service that the client request to access is one of the signals to decide whether the activity is normal or not. Not only in the IT system, but in all attacks, the focus has always been on the target's weak points. Based on this idea, if we examine dataset, it can be seen that the types of service accounts are *system, user and computer* accounts. Considering these three account types, it is clear that problems such as the weak password and encryption algorithm of the service that works with the *user account* will be prone to errors. From this perspective, the *system service account* types are defined as *acc_system*, the services that run with the *user account* are defined as *acc_personal*, and the services that run with the *computer account* are defined as *acc_computer*. Table 6 shows the volume and proportion of account types running the services in the data set.

Table 6: Account Type of Services

	Count	**Percent**
acc_system	940	16%
acc_user	130	2%
acc_computer	4924	82%

The another point that should be considered in the feature engineering section is determining the service type. Due to some services have been used for many years without update and then remains lagged behind with current technology, security vulnerabilities may emerge over time. This issue is generally application-based rather than operating system. Because user has become accustomed to some applications and cannot give up the old habits that is used for a long time. From that perspective, service types are divided into two: *system-based (svc_kerb)* and *application-based (svc_app)*. Since the system service related to our study is only Kerberos service, it was preferred to name it *svc_kerb*.

Perhaps the most obvious clues to determine whether a service account has been attacked is to examine service requests with weak encryption algorithms. It is also possible to compromise service accounts with a strong encryption algorithm, but it requires a very long time and powerful hardware to decrypt. As a result, features were added as *ticket_secure* and *ticket_unsecure*.

In addition to the number of service requests, variety of requested services and the ratio needs to be reflect statistically. To satify this parameter, *svc_req_count*, *dist_svc_count* and *svc_count_ratio* features have been added.

The *last_time_dif, total_time_dif, time_max, time_min and time_avg* features are going to be used to calculate the rate for service requests and to detect anomalies regarding service requests within a certain time period. The *issuspicious* feature was also used to determine and label whether the activity is malicous or not.

4.3.3.2 Feature Selection

The process of selecting a subset of features from the original features in order to minimize the feature space as much as possible while meeting predetermined criteria is known as feature selection. In the previous section, the study on how to convert raw data for feature engineering to be used in the ML model in order to detect the Kerberoasting attack was explained. Algorithms for feature selection fall into three broad categories: filter, wrapper and embedding methods [75].

In machine learning, feature selection has following roles:

1. Minimizes the feature space's dimension.
2. Accelerates an algorithm for learning.
3. Increases a classification algorithm's accuracy.
4. Boost the learning outcome's comprehensibility.

Since the aim of the thesis is to detect attacks and the attack contains several different features to achieve this goal, given that it evaluates every possible combination of features, the wrapper method was chosen for feature selection in our scenario.

To learn more about the dataset and find trends that can improve model performance, exploratory data analysis, or EDA, is used. In order to comprehend the distribution of both benign and malicious instances, this procedure required examining important elements from Event ID 4769, with an emphasis on the 74 produced Kerberoasting attacks. To investigate feature correlations and spot any outliers or abnormalities, visualizations including correlation matrices were employed as seen in Figure 21. In order to investigate feature variability, trends, and distributions, summary statistics were also computed. This information influenced the feature engineering procedure and assisted in directing the selection of specific characteristics for training machine learning models. The structure of the data was better understood thanks to this exhaustive EDA process, which also helped identify key factors that contribute to the detection of Kerberoasting attacks.

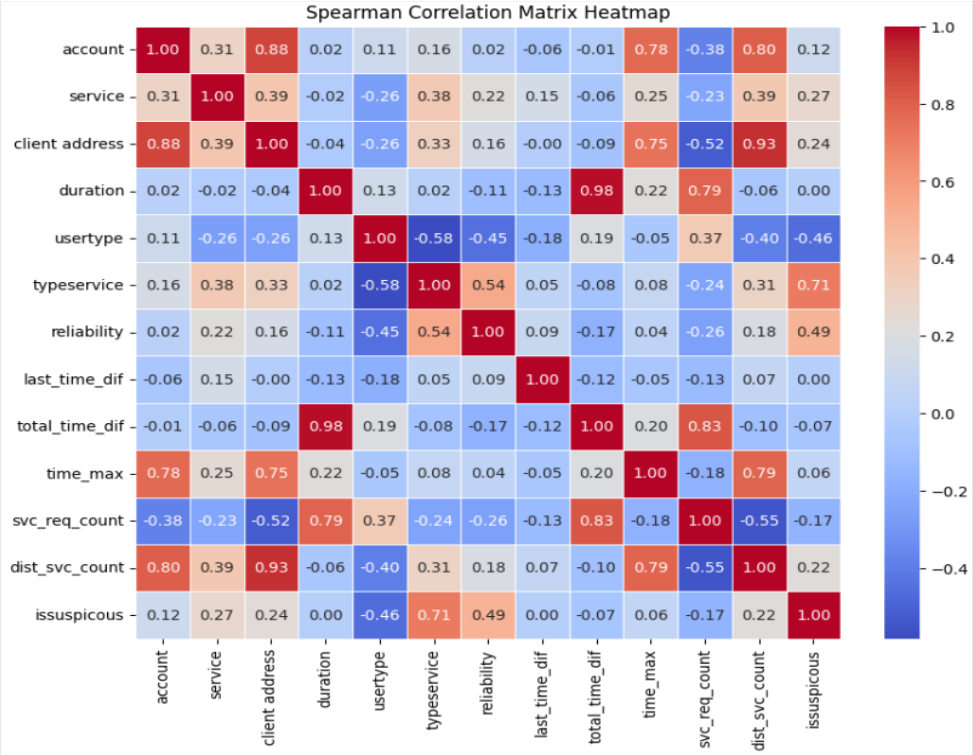

Figure 21: Spearman Correlation Matrix Heatmap

The one hot encoding approach was used to transform categorical variables into numerical values in order to prepare the dataset for machine learning and make sure it was in a format that could be used for model training. This change improved the model's capacity to discriminate between benign and malicious activity by allowing it to handle characteristics such as account and service names as binary vectors.

Given the mixture of categorical and continuous features in our dataset, Spearman correlation emerges as the most appropriate method for analyzing relationships. It effectively handles both types of variables and identifies associations even in cases of nonlinearity. The applicability of Spearman correlation to datasets with mixed variable types has been extensively discussed in the literature [76], further validating its suitability for our analysis. Figure 21 illustrates the correlation heatmap generated using the Spearman model, highlighting the relationships among the features in our dataset that are useful for detecting Kerberoasting attacks with supervised machine learning algorithms. *Total_time_dif* (cumulative total time difference between the last TGS requests) and *duration* (experiment time span) exhibit a very high correlation (0.98). This suggests that cumulative timing patterns during service ticket requests tend to grow proportionally with the overall experiment duration, providing a potential indicator of prolonged account activities. *Dist_svc_count* (distinct service request count) strongly correlates with account types (0.80), indicating that certain accounts, such as service accounts, generate a wider variety of service requests, a potential behavior associated with Kerberoasting attacks. *Time_max* (maximum time difference between TGS requests) shows strong correlations with both *account* (0.78) and *duration* (0.79), suggesting that certain accounts or extended experiments involve significant idle periods, which could align with attacker-controlled actions. *Svc_req_count* (service request count) also correlates with *duration* (0.79), indicating increased service activity over time, a

pattern often seen in malicious attempts. Finally, *typeservice* (e.g., DHCP, Exchange, IIS services) correlates significantly with *issuspicuous* (malicious or benign activity) at 0.71, making it a key feature in identifying the misuse of specific services during Kerberoasting attack attempts.

Dataset was ready for machine learning modeling by utilizing One Hot Encoding and Correlation Matrix analysis. Exploratory Data Analysis Matrix is used to summarize and understand the main characteristics of the dataset as seen in Figure 22. The code block used in data analysis is given in detail in Appendix A.

	account	service	client address	duration	usertype	...	total_time_dif	time_max	svc_req_count	dist_svc_count	issuspicous
count	5994.000000	5994.000000	5994.00000	5994.000000	5994.000000	...	5994.000000	5994.000000	5994.000000	5994.000000	5994.000000
mean	3.720053	1.390390	4.46630	24525.738238	1.945279	...	24349.411078	2705.748081	1647.352186	2.721555	0.012346
std	1.358993	1.288028	0.81182	5396.329224	0.227455	...	5747.893018	1809.087130	1255.214958	1.499111	0.110432
min	1.000000	1.000000	4.00000	0.000000	1.000000	...	0.000000	0.000000	1.000000	1.000000	0.000000
25%	3.000000	1.000000	4.00000	25901.000000	2.000000	...	25898.250000	1715.000000	566.250000	2.000000	0.000000
50%	3.000000	1.000000	4.00000	26256.000000	2.000000	...	26245.000000	1715.000000	1315.500000	2.000000	0.000000
75%	5.000000	1.000000	5.00000	26604.000000	2.000000	...	26591.000000	5742.000000	2703.750000	4.000000	0.000000
max	6.000000	12.000000	7.00000	30026.000000	2.000000	...	30026.000000	5787.000000	4202.000000	12.000000	1.000000

[8 rows x 13 columns]

Missing Values:
Series([], dtype: int64)

Figure 22: Explanatory Data Analysis Matrix

As seen in Figure 23, statistical graphs are given for the purpose of obtaining information and analyzing the characteristics of the data in the dataset. In these graphs, the distribution of data for each feature is shown.

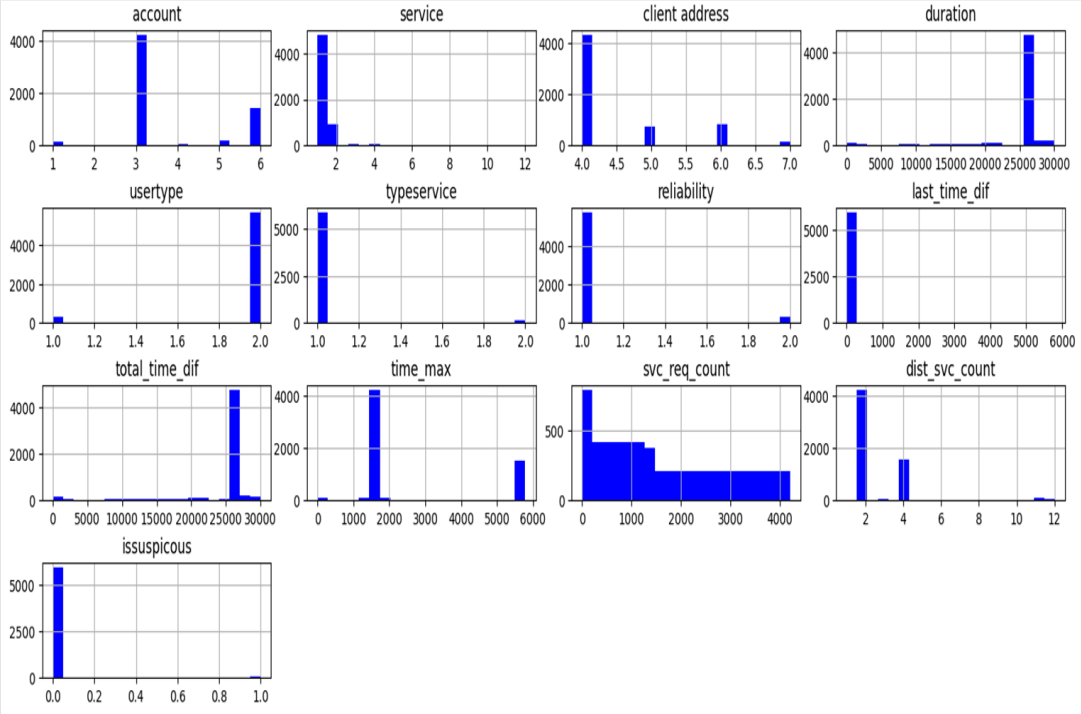

Figure 23: Explanatory Data Analysis Graphs

To determine which features contributed the most to detection, Feature Importance Scores were computed for a number of machine learning models. The Light GBM model demonstrated a notably higher value score for the *Service* attribute than the other eight models, as illustrated in Figure 24. It was challenging to understand the contributions of the remaining models because of this significant discrepancy, which caused their Feature Importance Scores to appear around zero.

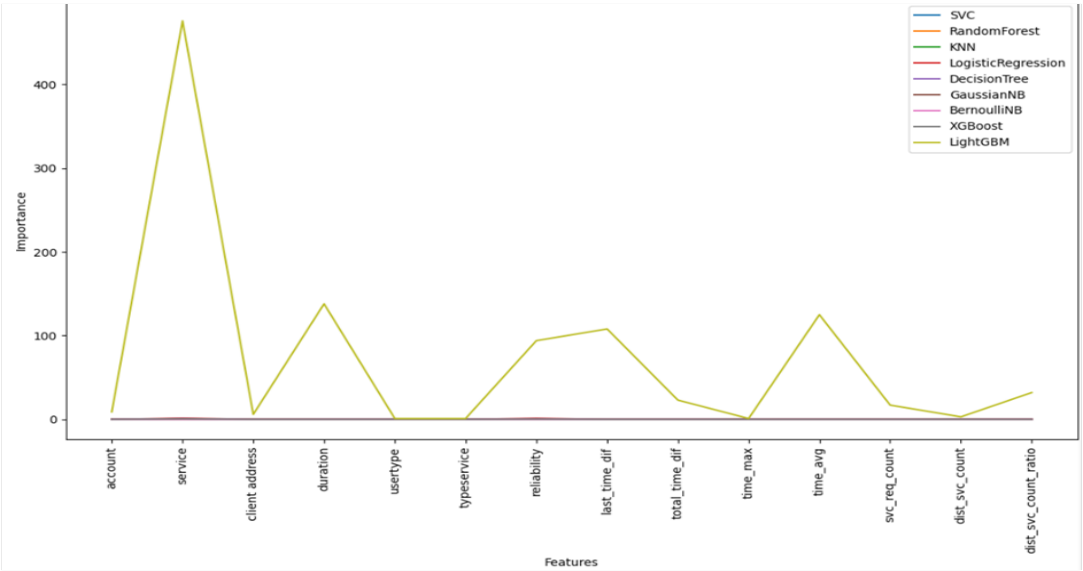

Figure 24: Feature Importance Scores with All Models

Light GBM was eliminated in order to produce a second Feature Importance Score graph Figure 24 that showed the performance of the remaining eight models. Light GBM's dominance in the *Service* feature had previously overshadowed the contributions of certain features, but this revision made it possible to compare the feature relevance ratings across different models in a more accessible manner.

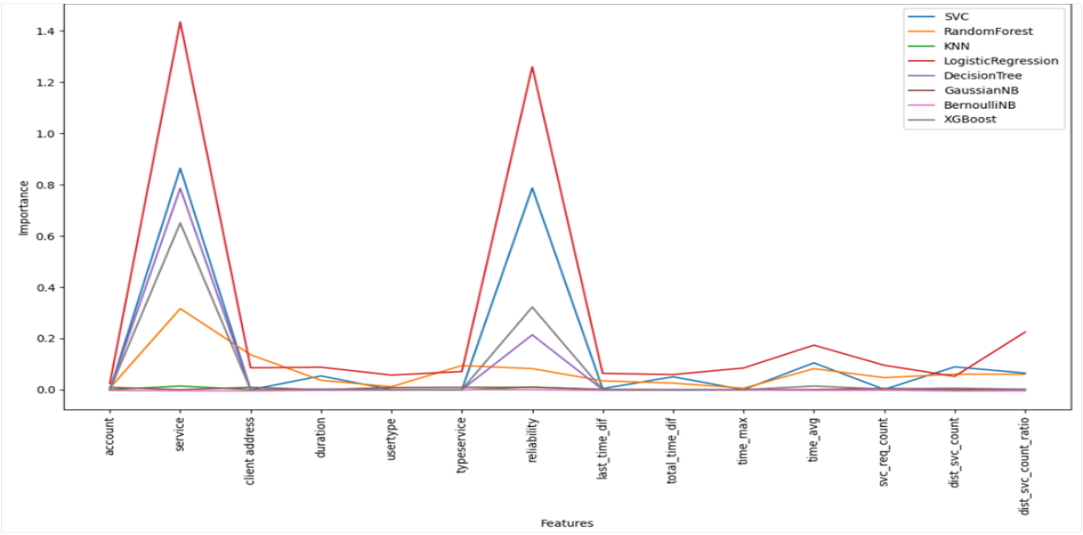

Figure 25: Feature Importance Scores Without Light GBM Model

4.3.4 Data Reduction

This section aims to improve the manageability of dataset without compromising the effectiveness of machine learning algorithm. The data reduction technique will further simplify and improve the readability of dataset without sacrificing its essential qualities. Feature selection, aggregation, and effective encoding are crucial phases in the data reduction phase for identifying Kerberoasting attacks. We may drastically lower dimensionality by concentrating on essential elements, such as *Account Name, Service Name, and Timestamp*, which are linked to Kerberoasting in Event ID 4769 logs. The most influential characteristics can be chosen with the aid of methods such as statistical filtering and recursive feature elimination [3].

Some features that have little predictive value is removed in order to further minimize the quantity of the data. Fields with static values, like some system-generated IDs or unnecessary metadata, like *Event Source, Event ID, Logon GUID, Client Port, Failure Code, Service ID, Account Domain* (where it merely identifies the log type), and non-informative timestamps that do not help capture behavioral patterns are removed. By eliminating these characteristics, the model may concentrate on characteristics that better capture the patterns of Kerberoasting attacks.

4.3.5 Data Splitting

Dividing the data into training, validation, and testing set is the last stage preparation for machine learning. Properly partitioning the data allows the machine learning model to be applied with greater accuracy when applied to new data.

Using a 70-30 or 80-20 ratio for training and test sets is standard procedure [77]. The model is trained on the training set and validated using with Stratified K-Fold Cross Validation technique. And evaluated on the three different testing datasets which have different malicious sample ratios. Machine learning models can be successfully validated using Stratified K-Fold Cross Validation, especially when applied to unbalanced datasets such as those used to identify Kerberoasting attacks. This method divides the dataset into k "folds" of similar size while maintaining the percentage of the target classes (normal vs. suspicious behavior, for example). For an appropriate assessment, particularly when the number of attacks is significantly lower than typical occurrences, this guarantees that each fold includes a representative sample of both attack and non-attack instances. We preferred the value 5 for k in our study. Aim is ensuring that every cluster accurately represents the whole data at this point. In our reference study, the malicious sample rate was 0.007. Therefore, we decided to use a malicious sample close to this rate in our study. Therefore, the dataset with a malicious sample rate of 0.012 was used in the training and validation dataset. Accordingly, in order to observe the performance of the model at varying rates, the performance of the models was tested by using datasets with malicious sample rates of 0.012, 0.008 and 0.004 in test datasets consisting of completely unseen data as seen in Table 7.

[3] Recursive Feature Elimination (RFE) is a feature selection technique used to optimize the performance of a machine learning model by selecting the most relevant features. Elimination process continues recursively, each time removing the least important feature, until a specified number of features are retained. RFE helps in reducing the complexity of the model, improving accuracy, and often reducing overfitting by focusing only on the most impactful features.

Table 7: Ratio of Malicious Samples

Dataset	Count	Malicious Ratio
Training & Validation	2997	1.2%
Test-1	2997	1.2%
Test-2	2997	0.8%
Test-3	2997	0.4%

4.4 Modelling

Finding patterns in data is the goal of machine learning, which can be done with supervised or unsupervised ML models. Regression, classification, forecasting, and clustering are examples of some ML types [78]. Experimental dataset is going to be used to identify the Kerberoasting attack in this thesis and it is going to be trained with a supervised machine learning model that is going to forecast possible Kerberoasting attack by using the available data. Ensuring the quality and accuracy of future predictions in new datasets heavily rely on accomplishment of this stage. At this step, important factors with high predictive values are tried to be determined in the most efficient way using machine learning techniques.

In order to assess the quality and validity of the dataset prepared for the attack detection as well as the effectiveness of the created models, the most suitable hyperparameter values are going to be utilized after applying GridSearch Cross Validation Tenchnique. Following discussing the models and parameters to be used in the first part, in-depth informations are going to be given about the models. The SIEM solution ELK Stack, allowed datasets to be exported as csv files. And then, this dataset might be used in SIEM solutions that has machine learning features, such as Splunk [79] or the models can be executed using Python codes via Jupyter Notebook. In order to use the dataset in machine learning models, the following two steps must be completed.

1. *Feature selection*
2. *Hyperparameter tuning*

The goal of the feature selection phase is to exclude irrelevant features from the dataset that barely affect the model's output. At this stage, the wrapper method was preferred. As a result of their flexibility in choosing features and ability to take into account different feature combinations, wrapper methods have the potential to identify optimal or nearly optimal subsets of features for a given model, which could improve model performance and result in simpler, easier-to-understand models.

In machine learning, hyperparameter tuning is crucial because it directly affects model performance through the optimization of parameters that influence the learning process. The model can prevent problems like overfitting or underfitting by determining the ideal hyperparameters, which will strike the correct balance between complexity and generalization. Hyperparameter tuning increases computing performance, makes the model more generalizable to new data, and customizes the model to the unique properties of the given task and dataset. Hyperparameter tuning guarantees that machine learning models reach optimal performance and maintains their competitiveness and efficacy in practical applications [80]. Parameters were preferred after applying GridSearch Cross Validation Technique for

all models to be applied regarding the Kerberoasting attack. Table 8 lists the applied hyperparameter values.

Table 8: Hyperparameter Grid and Best Hyperparameters for Machine Learning Models

Model	**Hyperparameter Grid**	**Best Hyperparameters**
Support Vector Classifier	{C: 0.1, 1, 10}, kernel: linear, rbf	{C: 10, kernel: linear}
Random Forest	{n_estimators: 50, 100}, max_depth: 10, 20	{max_depth: 20, n_estimators: 100}
K-Nearest Neighbors	{n_neighbors: 3, 5, 7}	{n_neighbors: 5}
Logistic Regression	{C: 0.1, 1, 10}	{C: 10}
Decision Tree Classifier	{max_depth: 5, 10, 15}	{max_depth: 5}
XGBoost	{learning_rate: 0.01, 0.1}, max_depth: 3, 5	{learning_rate: 0.1, max_depth: 3}
Light GBM	{learning_rate: 0.01, 0.1}, max_depth: 3, 5	{learning_rate: 0.01, max_depth: 3}
Gaussian Naive Bayes	No hyperparameters	{}
Bernoulli Naive Bayes	{alpha: 0.5, 1.0, 2.0}	{alpha: 0.5}

By methodically looking for the ideal set of hyperparameters, GridSearch Cross Validation is a crucial machine learning strategy that maximizes model performance. Applying GridSearch Cross Validation enables us to optimize performance and fine-tune the machine learning models in the context of detecting Kerberoasting attacks, where accuracy and detection precision are crucial. This is especially true when it comes to lowering false negatives and false positives, which are extremely sensitive in this security domain.

Configuration options known as hyperparameters control how machine learning algorithms behave. They have a big impact on the models' performance and are set before training rather than being immediately learned from the data. Random Forest's tree depth, XGBoost's learning rate, and Support Vector Machines' (SVM) penalty parameter are some examples. To make sure the model is neither underfitting nor overfitting the data, the ideal collection of hyperparameters must be determined.

In this thesis on detecting Kerberoasting attacks, GridSearch Cross Validation systematically evaluates multiple combinations of hyperparameter values. This process involves:

Grid Definition: We defined a grid of hyperparameters to test for each machine learning model. For example, in Random Forest, we may vary the number of trees (n_estimators), the maximum depth of the trees, and the minimum number of samples required to split an internal node. Similarly, for XGBoost, we might tune the learning rate, max_depth, and the number of boosting rounds.

Cross-Validation Process: GridSearch Cross Validation applies k-fold cross-validation to each combination of hyperparameters. This means the dataset is split into 'k' subsets, and the model is trained on 'k-1' subsets and validated on the remaining one. This process is repeated 'k' times to ensure the model is tested on all portions of the data, which helps prevent overfitting and ensures a robust evaluation of each hyperparameter combination.

Evaluation and Selection: For each set of hyperparameters, the model's performance is evaluated based on specified metrics, such as F2-score, precision, recall, or accuracy, depending on the model's performance goals. In our case, reducing false negatives (missed Kerberoasting attacks) is particularly critical, so metrics like recall and F2-score are given higher importance.

Best Hyperparameters: GridSearch Cross Validation identifies the hyperparameter combination that yields the highest performance on the chosen metric(s). The model is then retrained on the full training dataset using these optimal hyperparameters before final evaluation on unseen test data.

By determining the most efficient hyperparameters, GridSearch Cross Validation assists us in achieving the greatest model performance, which is essential for recognizing Kerberoasting attacks. Improved security and more successful detection in real-world situations result from our models' ability to differentiate between authentic service ticket requests and those that are indicative of Kerberoasting when hyperparameters are optimized.

This methodology guarantees that our models are optimized to manage the intricacies of Kerberoasting detection and helps to increase detection accuracy. In the following sections, a brief informations are going to be given about supervised machine learning models which are used in the study.

4.4.1 Gaussian Naive Bayes

Detecting Kerberoasting attacks with a Gaussian Naive Bayes model involves leveraging the algorithm's probabilistic framework to classify service requests as benign or indicative of Kerberoasting activity. The model assumes that features are conditionally independent given the class labels, allowing it to calculate the likelihood of observing a particular combination of feature values given each class. By estimating the parameters of Gaussian distributions for numerical features representing authentication attributes such as user account properties and authentication timestamps, and using categorical features such as service ticket characteristics, Gaussian Naive Bayes computes the probability of a given service request belonging to either a normal or Kerberoasting attack. This approach provides a simple yet effective means of distinguishing between legitimate and suspicious authentication behavior, enabling timely detection of potential Kerberoasting threats based on probabilistic reasoning [81].

4.4.2 Bernoulli Naive Bayes

By employing the probabilistic framework of the method to categorize authentication requests as either benign or suggestive of Kerberoasting activity, a Bernoulli Naive Bayes model can be used as a binary classification model. While Gaussian Naive Bayes implies that features follow a Gaussian distribution, Bernoulli Naive Bayes is designed for binary features, like flags that indicate whether a particular property is included in an service request or not. Bernoulli Naive Bayes determines the probability that an service request belongs to either a normal or the likelihood of witnessing each feature value given the class names. With this method, the model can recognize patterns and abnormalities in service request behavior, which makes it possible to detect possible Kerberoasting risks in a timely manner by analyzing the presence or lack of particular authentication features [82].

4.4.3 Decision Tree Classifier

Using a Decision Tree Classifier, one can use the algorithm's recursive feature space partition based on service request attributes to detect Kerberoasting attacks through the examination of attributes like service ticket details, authentication timestamps, and user account attributes, Decision Trees detect trends and deviations linked to Kerberoasting operations. The Decision Tree technique accurately classifies service requests as either legitimate or suspicious in regard to Kerberoasting attacks by using a set of binary decisions at each node to capture complicated correlations between attributes and the target variable. Security analysts can better grasp the decision-making process involved in recognizing potential dangers from Kerberoasting attacks by using decision trees, which provide transparent and easily interpretable rules for categorizing authentication requests [83].

4.4.4 Logistic Regression

When it comes to identifying Kerberoasting attacks, a sophisticated attack that targets Active Directory environments, Logistic Regression might be a useful tool. Here, service requests are classified as either unnoticeable or perhaps suggestive of Kerberoasting activities using Logistic Regression. By analyzing past data on authentication requests, the model is able to distinguish between suspicious activity that could be signs of a Kerberoasting attack attempt and patterns and abnormalities linked to normal user behavior. Logistic Regression analyzes variables such user account attributes, authentication time, and the requested service ticket's characteristics to help find minor anomalies from typical behavior that could indicate a Kerberoasting attack [15].

The interpretability and simplicity of Logistic Regression are important advantages for detecting malicious activities. Security analysts may interpret the model's predictions and take appropriate action by understanding the indicators of possible Kerberoasting activity. The model offers clear insights into the elements that drive its predictions. Additionally, Logistic Regression can manage the complexity of differentiating between malicious and legal authentication requests in an Active Directory context and is robust to noisy data. Nevertheless, the efficacy of Logistic Regression to identify Kerberoasting attacks depends on the availability of pertinent and high-quality variables as well as ongoing model improvement to accommodate changing attack strategies. Notwithstanding these difficulties, Logistic Regression would be important part of an all-encompassing cybersecurity plan that helps with the prompt identification and elimination of Kerberoasting attacks.

4.4.5 Support Vector Classifier

Because it can differentiate between legitimate and suspect authentication requests, the Support Vector Classifier (SVC), a supervised machine learning model, can be very useful in identifying Kerberoasting attacks. In order to maximize the gap between classes in this case, between malicious and benign login attempts linked to Kerberoasting SVC finds the best hyperplane [84]. For Kerberoasting detection, SVC is particularly useful because it can handle complex, high-dimensional data and is less prone to overfitting when regularized. The model's kernel trick enables it to map input features to higher dimensions, allowing it to detect complex patterns in Event ID 4769 logs, such as those indicative of ticket-granting service requests exploited by attackers. Despite its strengths, SVC's detection performance can be sensitive to the data scale and balance. With imbalanced datasets, it might require addi-

tional techniques to minimize false negatives, a critical focus for accurately identifying Kerberoasting attacks.

4.4.6 Random Forest Classifier

For classification tasks, the Random Forest Classifier is an ensemble machine learning model that constructs several decision trees during training and aggregates their results. Due to its resilience to overfitting and capacity to identify intricate, non-linear patterns in the data from Event ID 4769 characteristics, Random Forest can be very useful in identifying Kerberoasting attacks. In order to detect suspicious patterns, including unusual service ticket requests that are suggestive of Kerberoasting attempts, the model divides data into subsets according to feature values. Additionally, crucial indicators of attacks can be found using the Random Forest's built-in feature importance ranking, which is helpful for detection techniques and enhancing the interpretability of the model as a whole [85].

4.4.7 Extreme Gradient Boosting Machine (XGBoost)

Extreme Gradient Boosting, or XGBoost, is a popular machine learning model that uses gradient boosting techniques to increase prediction efficiency and accuracy. Because XGBoost can handle big datasets and high-dimensional feature spaces [86], like those from Event ID 4769, which are used to identify suspicious authentication operations, it is very good at identifying Kerberoasting attacks. In Kerberoasting detection, XGBoost has advantages for a number of reasons. First, because of its strong prediction capability, it may concentrate on cases that are challenging to categorize, facilitating in spotting subtle patterns in attack behaviors. Second, it offers tools for handling unbalanced data, such as the scale_pos_weight option, which can help deal with the problem of malicious events occurring infrequently. Third, XGBoost can handle huge datasets because of its parallel processing capabilities and effective memory usage, which shorten training times. Furthermore, XGBoost offers feature importance scores that assist in determining which features are the closest to the detection of Kerberoasting attacks. However, XGBoost requires careful tuning of its parameters (such as max depth, learning rate, and regularization settings) to prevent overfitting, as Kerberoasting data can be complex.

4.4.8 Light Gradient Boosting Machine (Light GBM)

A common gradient-boosting machine learning model for classification tasks, Light GBM is also employed in cybersecurity applications such as identifying Kerberoasting attacks. Because of its ability to manage big datasets and intricate feature structures, Light GBM can identify patterns in network logs or event data that correspond to threat signatures [87]. Light GBM has an advantage in terms of computational performance and memory use for Kerberoasting detection because it can split trees leaf-wise rather than level-wise. It can handle high-dimensional data from Event ID 4769 thanks to this method, where different aspects can help identify attack patterns. Support for regularization methods and categorical characteristics in the model further lessens overfitting, enhancing generalizability to novel attack patterns.

4.4.9 K-Nearest Neighbors (KNN)

By categorizing instances according to their proximity to labeled samples in the feature space, the K-Nearest Neighbors (KNN) algorithm is a non-parametric, instance-based learning model that can be useful for identifying Kerberoasting attacks. Each data point is evaluated by KNN by determining its distance from known examples of benign and malicious (Kerberoasting) activities, typically using the Manhattan or Euclidean distance. The performance of KNN for Kerberoasting detection is primarily determined by the feature scaling because of distance sensitivity and the choice of K (number of neighbors). The model may be less able to detect intricate attack patterns with a larger K, which could result in more false negatives even while it helps smooth out predictions [88]. Conversely, a lower K could make it harder to differentiate false positive results. KNN is easy to understand and straightforward, but because it must look through every instance for categorization, it can be computationally costly, particularly in big Kerberoasting datasets. KNN can attain a fair balance between detecting attacks (true positives) and minimizing false positives for efficient Kerberoasting detection by optimizing K and employing suitable feature engineering.

4.5 Evaluation

During the evaluation stage, the already constructed models are tested with test data. The crucial factor is assessing if they meet the goal, and a choice must be made on how to ultimately utilize the findings. Different usage scenarios can lead to variations in evaluation methods and criteria. Objective in this thesis is to assess how well supervised machine learning techniques can detect Kerberoasting attacks with minimum false alarm ratio. Thus, the designed supervised ML models are going to be evaluated on the test dataset in order to compare their performances.

Security monitoring seeks a binary classification answer; activity is suspicious or not. It is designed to generate an alarm if suspicious. A concept commonly used to evaluate classifiers is the confusion matrix. The confusion matrix provides a simple tabular summary of the predictions made by an ML model, allowing a simple evaluation of its performance. In binary classification, the matrix has two dimensions that compare predicted and actual values for two classes [89]. The terminology derived from the concept of the confusion matrix is directly relevant to security monitoring and is commonly utilized in real-world scenarios. The utilization of Figure 26 is depicted for determining malicious or benign activity, which can be interpreted in the following way:

- **True Positive (TP) :** A malicious activity that triggers an alert,
- **False Positive (FP) :** Alert is produced, but there is no malicious activity,
- **False Negative (FN) :** There is a malicious activity, but no alert triggers,
- **True Negative (TN) :** There is no malicious activity and no alert is produced.

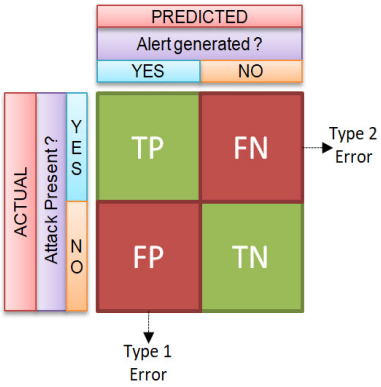

Figure 26: Kerberoasting Attack Detection Confusion Matrix

A confusion matrix is a table used to summarize the performance of a classification model. Performance is generally evaluated with 3 parameters. To briefly touch upon this issue [90];

Accuracy : This metric accurately forecasts the number of each class, both positive and negative. The highest level of accuracy is desirable.

$$\text{ACC} = \frac{\text{True Positives + True Negatives}}{\text{Sum of All Observations}} = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}}$$

Accuracy generally has one major drawback; it hides the issue of class inequality. Data scientists frequently utilize precision and recall as classification metrics to maximize model performance. They provide us with information about model performance that accuracy cannot.

Precision : One measure of the quality of positive predictions is precision. When we intend to reduce false positives, we apply precision.

$$\text{PRE} = \frac{\text{True Positives}}{\text{All Actual Positives}} = \frac{\text{TP}}{\text{TP + FP}}$$

Recall : Recall provides information about how well the model detects true positives. Recall is used when reducing false negatives is crucial.

$$\text{REC} = \frac{\text{True Positives}}{\text{All Predicted Positives}} = \frac{\text{TP}}{\text{TP + FN}}$$

F Score : When eliminating false positives and negatives requires a trade-off, we use the F-score to determine a balance between precision and recall. If a parameter $\beta = 1$ is used, there needs to be equality between Recall and Precision. Given that the thesis seeks to identify Kerberoasting attacks, a comparison of type 1 and type 2 errors must be made. In conclusion, false negative errors are Type 1 errors, whereas false positive errors are Type 2 errors. When it comes to detect Kerberoasting attack, Type 1 errors are more significant since the inability to identify an actual attack is more significant than the alert raised for an imaginary attack. Recall is hence has more importance in this area. In order to favor the Recall parameter more arbitrary, we should interpret the F score as 2, the number 1.0 should be used if we were to choose Precision from perspective of detection.

$$F_\beta = (1 + \beta^2) \times \frac{\text{PRE x REC}}{\beta^2 \text{ x PRE + REC}}$$

The dataset is divided into many folds (in this case, five equal sized folds) using stratified K-Fold Cross Validation Technique. Only a portion of the dataset is used as a test set in each iteration, and each fold is utilized as a test set only once, resulting in metrics being reported for a single fold at a time. We computed the average of the metrics over all K folds to view metrics across all the folds. This determines and presents the average performance for each of the five folds as seen in Table 9. According to the result; SVC, Logistic Regression, and XGBoost offer a good balance between performance and consistency. In contrast, models like Random Forest and LightGBM show higher variability and may not be suitable for scenarios requiring high predictability without further optimization.

Table 9: Results for K-Fold Stratified Cross Validation (Average)

Model	**TN**	**FP**	**FN**	**TP**	**Precision**	**Recall**	**F1**	**F2**	**F2 StdDev**	**Accuracy**
SVC	590.8	1.2	1.2	6.2	0.861429	0.835714	0.834105	0.832540	0.114011	0.995997
Logistic Reg.	590.6	1.4	1.2	6.2	0.841984	0.835714	0.822439	0.827284	0.107893	0.995663
XGBoost	590.4	1.6	1.2	6.2	0.832828	0.835714	0.814490	0.823134	0.109589	0.995329
GaussianNB	583.0	9.0	0.0	7.4	0.467172	1.000000	0.632702	0.808655	0.050031	0.984987
DecisionTree	590.2	1.8	1.4	6.0	0.772222	0.810714	0.784744	0.798486	0.125095	0.994661
KNN	591.0	1.0	1.6	5.8	0.895556	0.782143	0.814105	0.790266	0.104377	0.995663
Random Forest	590.4	1.6	1.6	5.8	0.805556	0.785714	0.780455	0.780856	0.132253	0.994662
LightGBM	591.0	1.0	1.8	5.6	0.876984	0.753571	0.771984	0.756772	0.221357	0.995328
BernoulliNB	574.6	17.4	0.0	7.4	0.319545	1.000000	0.477986	0.689321	0.079097	0.970973

In this thesis, we also evaluated the performance of machine learning models for Kerberoasting attack detection using unseen datasets with varying ratios of malicious samples: 0.012, 0.008, and 0.004. These ratios were selected based on a reference study that utilized a ratio of 0.007. By choosing similar, yet slightly varied ratios, we aimed to ensure that our results could be effectively compared with the findings of the reference study. This approach allowed us to assess the robustness and generalizability of the models in detecting Kerberoasting attacks under different levels of class imbalance, a critical aspect given that malicious events often constitute a small fraction of network traffic. Evaluating the models with datasets that simulate realistic attack scenarios where the prevalence of malicious activity is low provides a more comprehensive understanding of their performance.

While there is no universally accepted ratio for malicious samples specific to Kerberoasting attack detection, the principle of testing machine learning models with imbalanced datasets is widely recognized in cybersecurity research. For instance, a survey [91] emphasizes the challenges associated with imbalanced datasets in anomaly detection, a key area within cybersecurity. It highlights the need for specialized validation techniques, such as stratified cross-validation, to ensure that models can effectively identify rare but critical events, like security breaches and other cyber attacks. The ratios we employed mirror real-world conditions where attacks are infrequent, highlighting the models' sensitivity and adaptability to detecting rare events. By comparing model performance across various ratios, we were able to examine their ability to maintain detection metrics such as accuracy, precision, and recall in scenarios that reflect practical production environments. This testing strategy demonstrates a thoughtful consideration of the dataset characteristics, which is essential for assessing model effectiveness in operational settings where detecting rare but critical events is paramount.

As seen in Table 10 and Figure 27; after applying the models using the first test dataset with a malicious sample ratio of 0.012, we observed that the results aligned well with the outcomes from the K-Fold Stratified Cross Validation (Average). Logistic Regression, which performed consistently well during cross-validation, maintained high recall (0.972973) and a balanced F1 score (0.765957) on this dataset, demonstrating its reliability across different sample ratios. Interestingly, Random Forest, which had a slightly lower recall (0.810714) in cross-validation, showed an improvement in precision (0.882353) and overall performance when tested with unseen data, yielding a high F1 score of 0.845070 and accuracy of 0.996330. Gaussian Naive Bayes, while achieving perfect recall in both cross-validation and test phases, exhibited low precision (0.440476) in the 0.012 sample ratio dataset, a sharp contrast to its performance in the stratified cross-validation, where it similarly struggled with precision but maintained better overall balance.

Table 10: Results for 0.012 Ratio of Malicious Samples

Model	TN	FP	FN	TP	Precision	Recall	F1	F2	Accuracy
LogisticRegression	2939.0	21.0	1.0	36.0	0.631579	0.972973	0.765957	0.878049	0.992659
RandomForest	2950.0	10.0	7.0	30.0	0.750000	0.810811	0.779221	0.797872	0.994328
GaussianNB	2913.0	47.0	0.0	37.0	0.440476	1.000000	0.611570	0.797414	0.984318
XGBoost	2949.0	11.0	7.0	30.0	0.731707	0.810811	0.769231	0.793651	0.993994
DecisionTree	2949.0	11.0	7.0	30.0	0.731707	0.810811	0.769231	0.793651	0.993994
KNN	2947.0	13.0	7.0	30.0	0.697674	0.810811	0.750000	0.785340	0.993327
SVC	2909.0	51.0	0.0	37.0	0.420455	1.000000	0.592000	0.783898	0.982983
BernoulliNB	2902.0	58.0	0.0	37.0	0.389474	1.000000	0.560606	0.761317	0.980647
LightGBM	2957.0	3.0	30.0	7.0	0.700000	0.189189	0.297872	0.221519	0.988980

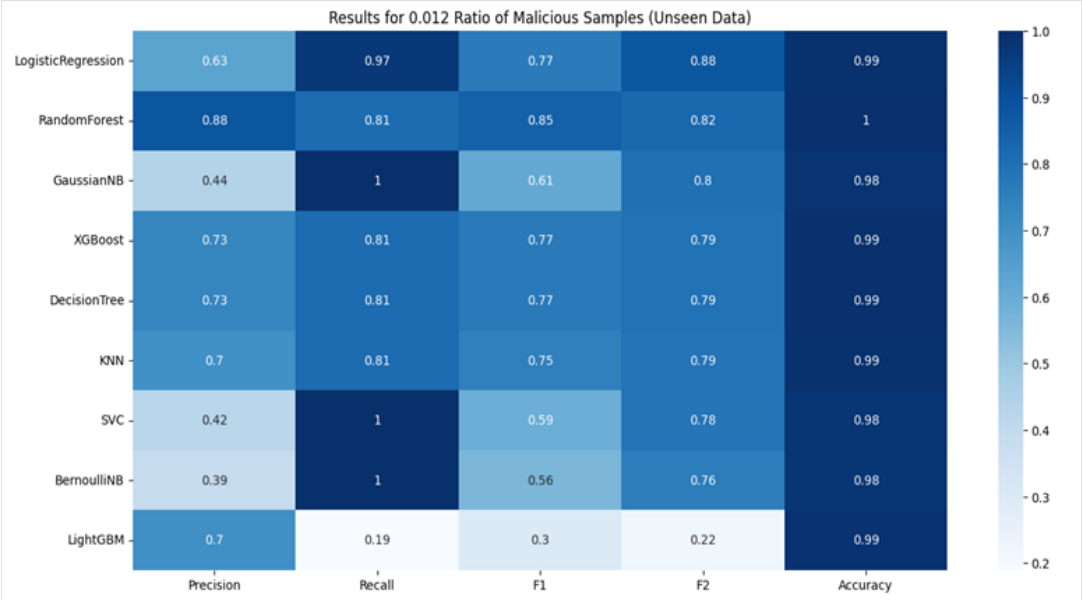

Figure 27: Result Graph for Test Dataset-1 with 0.012 Malicious Samples Ratio

As seen in Table 11 and Figure 28; with the second dataset, having a malicious sample ratio of 0.008, we noticed more divergence in the results. Random Forest retained its strong performance, showing precision of 0.294118 and recall of 0.833333, but the F1 score (0.434783) was lower compared to the 0.012 dataset. Logistic Regression, which achieved a perfect recall of 1.0, struggled significantly

in terms of precision (0.210526), resulting in a lower F1 score (0.347826). This decline was notable compared to its more stable performance with the 0.012 dataset, indicating sensitivity to the further reduction in malicious samples. GaussianNB and BernoulliNB, which exhibited perfect recall in both cross-validation and the first test dataset, continued to face precision challenges, significantly affecting their overall F1 scores.

Table 11: Results for 0.008 Ratio of Malicious Samples

Model	TN	FP	FN	TP	Precision	Recall	F1	F2	Accuracy
LogisticRegression	2940.0	45.0	0.0	12.0	0.210526	1.000000	0.347826	0.571429	0.984985
RandomForest	2955.0	30.0	2.0	10.0	0.250000	0.833333	0.384615	0.568182	0.989323
DecisionTree	2954.0	31.0	2.0	10.0	0.243902	0.833333	0.377358	0.561798	0.988989
XGBoost	2954.0	31.0	2.0	10.0	0.243902	0.833333	0.377358	0.561798	0.988989
KNN	2952.0	33.0	2.0	10.0	0.232558	0.833333	0.363636	0.549451	0.988322
GaussianNB	2913.0	72.0	0.0	12.0	0.142857	1.000000	0.250000	0.454545	0.975976
SVC	2909.0	76.0	0.0	12.0	0.136364	1.000000	0.240000	0.441176	0.974641
BernoulliNB	2902.0	83.0	0.0	12.0	0.126316	1.000000	0.224299	0.419580	0.972306
LightGBM	2978.0	7.0	9.0	3.0	0.300000	0.250000	0.272727	0.258621	0.994661

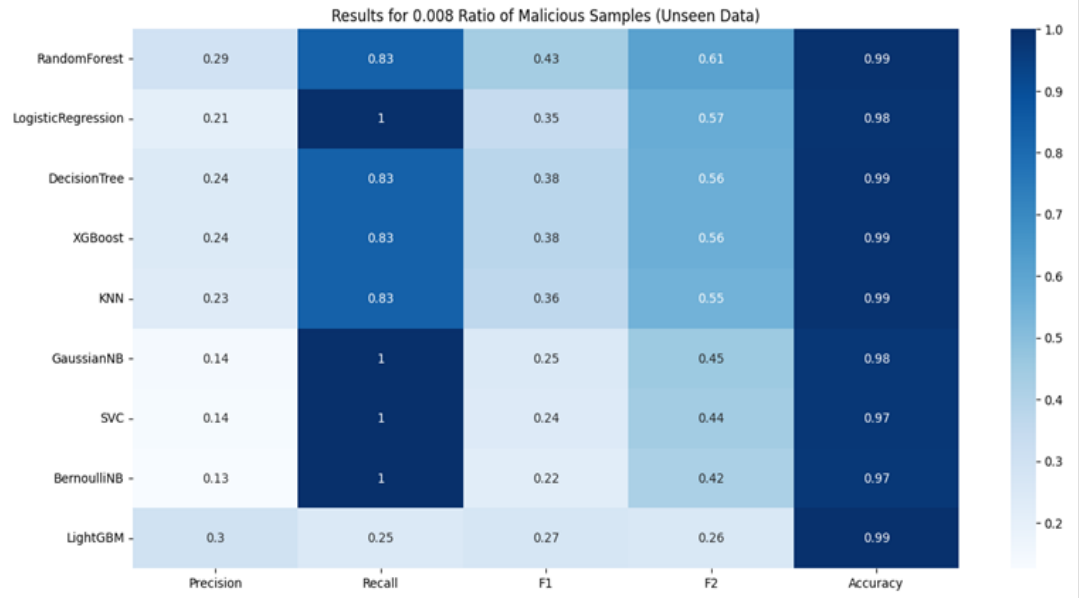

Figure 28: Result Graph for Test Dataset-2 with 0.008 Malicious Samples Ratio

As seen in Table 12 and Figure 29, in the dataset with the smallest ratio of malicious samples (0.004), we observed trends similar to the 0.008 ratio dataset, though some models showed a slight improvement in precision. For instance, Random Forest's precision remained steady at 0.294118 with the same recall of 0.833333, resulting in an unchanged F1 score. Logistic Regression continued to exhibit strong recall (1.0) but low precision (0.210526), similar to its performance on the 0.008 dataset. GaussianNB and BernoulliNB, once again, achieved perfect recall but were heavily impacted by poor precision, reflecting the difficulty of maintaining balanced performance across datasets with very few malicious samples. The code block used in model evaluation is given in detail in Appendix B.

Table 12: Results for 0.004 Ratio of Malicious Samples

Model	TN	FP	FN	TP	Precision	Recall	F1	F2	Accuracy
LogisticRegression	2940.0	45.0	0.0	12.0	0.210526	1.000000	0.347826	0.571429	0.984985
RandomForest	2955.0	30.0	2.0	10.0	0.250000	0.833333	0.384615	0.568182	0.989323
DecisionTree	2954.0	31.0	2.0	10.0	0.243902	0.833333	0.377358	0.561798	0.988989
XGBoost	2954.0	31.0	2.0	10.0	0.243902	0.833333	0.377358	0.561798	0.988989
KNN	2952.0	33.0	2.0	10.0	0.232558	0.833333	0.363636	0.549451	0.988322
GaussianNB	2913.0	72.0	0.0	12.0	0.142857	1.000000	0.250000	0.454545	0.975976
SVC	2909.0	76.0	0.0	12.0	0.136364	1.000000	0.240000	0.441176	0.974641
BernoulliNB	2902.0	83.0	0.0	12.0	0.126316	1.000000	0.224299	0.419580	0.972306
LightGBM	2978.0	7.0	9.0	3.0	0.300000	0.250000	0.272727	0.258621	0.994661

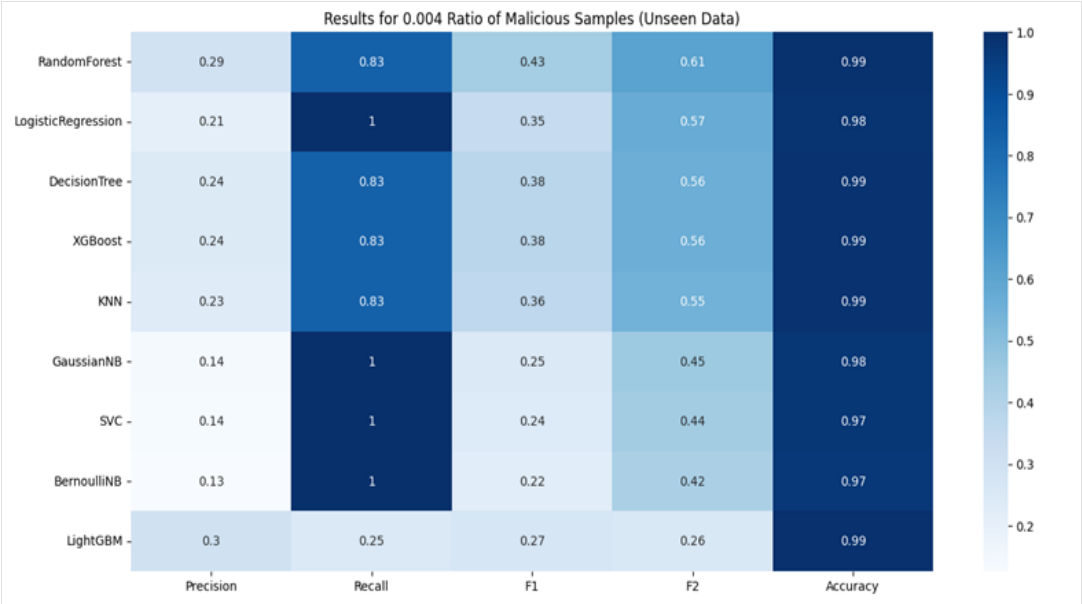

Figure 29: Result Graph for 0.004 Malicious Samples Ratio

Overall, the comparison between the K-Fold Stratified Cross Validation (Average) results and the different malicious sample ratios highlights that while Random Forest and Logistic Regression models, were able to maintain strong performance with unseen data, others, particularly Gaussian Naive Bayes and Bernoulli Naive Bayes, exhibited pronounced variability in precision when tested on imbalanced datasets. The Logistic Regression and Random Forest models that performed consistently well and demonstrated adaptability to varying sample ratios, confirming their suitability for deployment in production environments where malicious events are rare. Support Vector Classifier and Random Forest Classifier models gave consistent results depending on the data. However, we can express that Logistic Regression is the model that gives the best results in all datasets regardless of the data. Similarly, Bernoulli Naive Bayes and Light Gradient Boosting Machine models gave the worst results in terms of detecting Kerberoasting attack regardless of the data.

CHAPTER 5

RESULTS

5.1 Discussion

In this section, we are going to compare the performance results of our machine learning models for detecting Kerberoasting attacks with those reported in the study by Kotlaba et al. [15]. This comparative analysis allowed us to evaluate the relative effectiveness of our approach in relation to established benchmarks. The main theme of referance study was to compare the ability of supervised, semi-supervised and unsupervised ML algorithms to detect Kerberoasting attack and to reduce false positive results compared to signature recognition methods by using ML algorithms. In the study, they were able to reduce the number of false positive alarms, but at this point we wanted to draw attention to another issue that we think has been overlooked. In particular, false negative results pose more danger in the detection of Kerberoasting attacks. Because a real attack that does not produce an alarm that can be overlooked can have worse results than a false alarm of an attack that does not actually exist. For this reason, we tried to identify another supervised machine learning algorithms that also reduces false negative results in the study. In this thesis, we also applied other supervised ML algorithms to own dataset, considering that each environment is going to have its own characteristics and that the time spent in the training phase of machine learning may have dire consequences in terms of domain compromising. Our main research question was whether are there any other supervised ML algorithms that can reduce false negative results too. In the referance study, Support Vector Machine and Random Tree Classifier algorithms were used as supervised ML algorithms as seen in Table 13 which represented 22 real Kerberorasting attacks.

Table 13: Comparison of ML algorithms in terms of Kerberoasting Detection

Results	**Threshold Rule**	**Support Vector Classifier**	**Random Forest Classifier**
True Positive	20	13	18
False Negative	2	9	4
False Positive	73	0	0
True Negative	3030	3103	3103
Precision	0.215	1.000	1.00
Recall	0.909	0.591	0.818
F1 Score	0.348	0.743	0.900

The previous study aimed to reduce the high false positive rate that resulted from using the Threshold Rule. By implementing machine learning models, such as the Support Vector Classifier (SVC) and

Random Forest Classifier, they were able to significantly decrease the FP rate, with both models yielding zero false positives. However, while they succeeded in lowering FPs, they struggled with FNs, particularly in the case of SVC, which had a lower recall, indicating missed Kerberoasting attacks.

In our study, we placed a stronger emphasis on minimizing both false positives and false negatives, as we concluded that missing an attack (FN) is more harmful than a false alarm (FP). After testing various supervised ML models, we confirmed that Random Forest Classifier, as seen in the previous study, remain among the most successful models for detecting Kerberoasting attacks. However, we discovered that one of our models Logistic Regression produced better results than the SVC. While the Random Forest Classifier continued to perform well, especially in terms of maintaining a good balance between precision and recall, it was not surpassed. Logistic Regression, on the other hand, outperformed SVC by achieving a higher recall, meaning it detected more attacks, albeit with a slight trade-off in precision. These findings highlight that while Random Forest remains robust, alternative models like Logistic Regression can be more effective than SVC for reducing both FNs and FPs.

However, we observed that the performance of the Random Forest Classifier and SVC models in the reference study may vary depending on the dataset. According to our results, we can say that the Logistic Regression model provides better results in detecting Kerberoasting attacks regardless of the dataset. Similarly, we can say that Bernoulli Naive Bayes and Light GBM models are among the algorithms that give the worst results in detecting Kerberoasting attacks regardless of the dataset.

5.2 Contributions

It was not possible to provide AD audit data from prior studies due to sensitive data in the event logs that utilized for analysis may potentially reveal IP addresses of servers, users, and other similar details about objects. Thus, there is not any publicly available dataset. Those who wish study in this subject have to decide whether to recreate a virtual environment or deal with the issue of being unable to export the data from real IT environment. Due to these reasons, the study was conducted virtually and was not concerned with data confidentiality. So it provided opportunity to determine capability of supervised machine learning algorithms to identify Kerberoasting attacks. The dataset and code generated during this study has been made publicly available.

We have come to a reliable conclusion regarding how the models behave when detecting Kerberoasting attacks. Logistic Regression is therefore the best model, independent of the dataset. No matter the dataset, the Bernoulli Naive Bayes and Light Gradient Boosting Machine models performed the poorest when it came to detecting Kerberoasting attacks.

5.3 Future Work

This thesis can continue by applying the suggested detection criteria to different SIEM solutions that has supervised machine learning capability. Real-world AD environments can be used to confirm the capability of the suggested algorithms and actual attacks might conducted to assess its detection capabilities. As this thesis uses a static subset of log data, it is necessary to continuously observe the behavior of solutions and performance in real AD structures continually.

The applicability of alternative machine learning methods, particularly those that are semi-supervised or unsupervised, for identifying attacks can be investigated from a research standpoint to see whether they can lower the number of false negatives too. When compared to the outcomes of previous investigations, supervised ML algorithms approaches did fairly well. The dataset imbalance present in the attack detection however, can still have a negative effect on the outcomes. So this can be another study topic in this area.

5.4 Limitations

Becasue of the resource availability, there was a fairly strict limit on the number of users, clients, servers, and services that could be established in the artificial virtual environment used for this study. In spite of this, in just three weeks, a sizable data set was obtained. It is not possible to predict what kind of consequences may arise from imbalances in very large data sets.

CHAPTER 6

CONCLUSION

In the thesis, a method shown to detect the Kerberoasting attack, which is one of the attacks targeting the Microsoft Active Directory basic authentication mechanism Kerberos protocol, by using supervised machine learning algorithms and Windows security logs. The aim was to analyze the algorithms that provide the most efficient results to detect Kerberoasting attack among the most popular supervised machine learning algorithms.

After a preliminary introduction to cryptography terminology, Active Directory technology, authentication and authorization concept and detection methods, an analysis of the Kerberosting attack detection regarding Active Directory was carried out, focusing on machine learning algorithms suitable for detection. Unlike previous studies on detecting the attack, the aim was to minimize false negative results as well as false positive detections with supervised machine learning algorithms. So, the study made use of supervised machine learning algorithms, since the every IT environment has unique characteristics of its own. Another reason for choosing supervised machine learning is algorithm also needed to pick up on these distinct features of the surroundings for the IT environment without losing time. Supervised machine learning algorithms were selected in preference to unsupervised and semi-supervised due to reason that an overlooking Kerberoasting attack can cause huge costs that stems from domain compromise. It has been tried to determine whether is there any supervised machine learning algorithm that can accurately detect Kerberoasting attacks without false positives and negatives. To achieve this goal, a virtual laboratory environment been established and then performed Kerberoasting attacks on this environment. Event logs of the attacks have been sent to the ELK Stack SIEM environment, which was also deployed in the cloud. The dataset was then exported from the SIEM tool to be used as input in the machine learning algorithms.

Rate of the dataset used to train the algorithm in supervised machine learning algorithm has an impact on this result. Nevertheless, best rates and parameters were applied in order to limit the impact of this problem on the outcomes after employing Gridsearch Cross Validation Technique. Because, the goal was to find out how well the algorithm detects Kerberoasting attacks. Previous research were unable to make public datasets available due to the sensitivity of Active Directory (AD) audit data because event logs contain potentially revealing information such as user details and server IPs. The only options available to researchers are to either develop virtual environments or deal with limitations on exporting data from operational IT systems. The tests in this study were carried out in a virtual environment, which allowed us to assess the efficacy of supervised machine learning algorithms to detect Kerberoasting attacks. And also it is free from worries about data confidentiality. The dataset and codes are publicly available. Our results show that, in Kerberoasting detection without dataset dependence, Logistic Regression consistently performed better than alternative models. On the other

hand, LightGBM and Bernoulli Naive Bayes produced the poorest outcomes. We also saw that the Random Forest Classifier and Support Vector Classifier's performance differed based on the dataset, demonstrating how data properties affect how well these algorithms work.

The quality of the algorithms utilized in this thesis can be demonstrated by doing extensive testing in real scenarios with different operating system versions and applying varied configurations. Other areas are also available for development area in the future. Furthermore, there is not a study that targets multi-site or multi-forest AD samples. New attack tools and strategies are continually being created, and security threats are changing. In response, organizations must continually enhance their detection systems and put security measures in place.

REFERENCES

[1] B. Sheresh and D. Sheresh, "What is directory service?," in *Understanding directory services*, Sams Publishing, 2002.

[2] S. Krishnamoorthi and J. Carleton, "Active directory holds the keys to your kingdom, but is it secure?," *Frost & Sullivan, available at https://www.frost.com/growth-opportunity-news/active-directory-holds-the-keys-to-your-kingdom-but-is-it-secure/*, vol. 20. [Online; accessed 21-December-2023].

[3] D. Francis, *Mastering Active Directory: Deploy and Secure Infrastructures with Active Directory, Windows Server 2016, and PowerShell*. Packt Publishing Ltd, 2019.

[4] Microsoft, "Five steps to securing your identity infrastructure," *Available at https://learn.microsoft.com/en-us/azure/security/fundamentals/steps-secure-identity*, Article: 10/12/2023. [Online; accessed 21-December-2023].

[5] B. P. Can Ozkan, Dave Singelée, "How insecure configurations of active directory could lead to full compromise of scada systems," *Available at https://cosicdatabase.esat.kuleuven.be/backend/publications/files/report/3728*. [Online; accessed 07 August 2024].

[6] D. Demers and H. Lee, "Kerberoasting: Case studies of an attack on a cryptographic authentication technology," *International Journal of Cybersecurity Intelligence & Cybercrime*, vol. 5, no. 2, pp. 25–39, 2022.

[7] MIT, "Kerberos documentation (1.21.3)," *Available at https://web.mit.edu/kerberos/krb5-latest/doc/*. [Online; accessed 03-December-2023].

[8] D. Guster, C. Hemminger, J. Meunier, and C. Schroeder, "Using kerberos to harden the active directory system (ldap) in a domain used to support grid/clustering activity,"

[9] B. I. Mokhtar, A. D. Jurcut, M. S. ElSayed, and M. A. Azer, "Active directory attacks—steps, types, and signatures," *Electronics*, vol. 11, no. 16, p. 2629, 2022.

[10] P. Löfgren, "Active directory tiering," *Available at https://www.truesec.com/security/active-directory-tiering*, 2024. [Online; accessed 31 July 2024].

[11] F. Jeannot, "Kerberos," *V5 Montréal, Canada, July 2023, Y760, v1.0 Available at https://franckybox.com/wp-content/uploads/kerberosv5.pdf*, 2023.

[12] S. M. Bellovin and M. Merritt, "Limitations of the kerberos authentication system," in *Proceedings of the Usenix Winter 1991 Conference, Dallas, TX, USA, January 1991*, pp. 253–268, USENIX Association, 1991.

[13] C. D. Motero, J. R. B. Higuera, J. B. Higuera, J. A. S. Montalvo, and N. G. Gómez, "On attacking kerberos authentication protocol in windows active directory services: A practical survey," *IEEE Access*, vol. 9, pp. 109289–109319, 2021.

[14] L. Kotlaba, S. Buchovecká, and R. Lórencz, "Active directory kerberoasting attack: Monitoring and detection techniques.," in *ICISSP*, pp. 432–439, 2020.

[15] L. Kotlaba, S. Buchovecká, and R. Lórencz, "Active directory kerberoasting attack: Detection using machine learning techniques.," in *ICISSP*, pp. 376–383, 2021.

[16] M. S. Elmastaş and C. Eyüpoğlu, "Detection of current attacks in active directory environment with log correlation methods.," *Journal of Aeronautics & Space Technologies/Havacilik ve Uzay Teknolojileri Dergisi*, 2023.

[17] R. Younisse, M. Alkasassbeh, M. Almseidin, and H. Abdi, "An early detection model for kerberoasting attacks and dataset labeling," *Jordanian Journal of Computers and Information Technology*, vol. 9, no. 1, 2023.

[18] J. Buchmann, *Introduction to cryptography*, vol. 335. Springer, 2004.

[19] J. Black, "Authenticated encryption," in *Encyclopedia of Cryptography and Security* (H. C. A. van Tilborg, ed.), Springer, 2005.

[20] M. Nelson, "56-bit des algorithm broken in record time," *Computers & Security*, vol. 2, no. 18, pp. 149–150, 1999.

[21] B. Kaduk and M. Short, "Deprecate triple-des (3DES) and RC4 in kerberos," *RFC*, vol. 8429, pp. 1–10, 2018.

[22] W. contributors, "Rsa (cryptosystem) — Wikipedia, the free encyclopedia," *Available at https://en.wikipedia.org/w/index.php?title=RSA$_{(}cryptosystem)oldid = 1189590610$, 2023.* $[Online; accessed\ 29 - December - 2023]$.

[23] Wikipedia contributors, "Elliptic-curve cryptography — Wikipedia, the free encyclopedia," *Available at https://en.wikipedia.org/w/index.php?title=Elliptic-curve$_{c}ryptographyoldid = 1189440768$, 2023.* $[Online; accessed\ 29 - December - 2023]$.

[24] Wikipedia contributors, "Diffie–hellman key exchange — Wikipedia, the free encyclopedia," *Available at https://en.wikipedia.org/w/index.php?id=1192060554*, 2023. [Online; accessed 29-December-2023].

[25] S. Debnath, A. Chattopadhyay, and S. Dutta, "Brief review on journey of secured hash algorithms," in *2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix)*, pp. 1–5, IEEE, 2017.

[26] Encryption Consulting LLC., "What is sha? what is sha used for?," *Available at https://www.encryptionconsulting.com/education-center/what-is-sha/*, 2024. [Online; accessed 21-February-2024].

[27] K. Yasar and A. Zola, "Active directory tiering," *Available at https://www.techtarget.com/searchdatamanagement/definition/hashing*, 2024. [Online; accessed 31 July 2024].

[28] GeekforGeeks, "What is the md5 algorithm?," *Available at https://www.geeksforgeeks.org/what-is-the-md5-algorithm/*, 2023. [Online; accessed 29-December-2023].

[29] A. Mohammed Ali and A. Kadhim Farhan, "A novel improvement with an effective expansion to enhance the md5 hash function for verification of a secure e-document," *IEEE Access*, vol. 8, pp. 80290–80304, 2020.

[30] JavaTpoint, "Sha algorithm in cryptography," *Available at https://www.javatpoint.com/sha-algorithm-in-cryptography?*, 2023. [Online; accessed 29-December-2023].

[31] Okta Corporation, "Hmac (hash-based message authentication codes) definition," *Available at https://www.okta.com/identity-101/hmac/*, 2023. [Online; accessed 29-December-2023].

[32] Microsoft, "Network security: Configure encryption types allowed for kerberos," *Available at https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-configure-encryption-types-allowed-for-kerberos*, 2023. [Online; accessed 29-December-2023].

[33] The CyberArk Corporation, "Directory services," *Available at https://www.cyberark.com/what-is/directory-services/*, 2023. [Online; accessed 22-November-2023].

[34] V. Kanade, "What is active directory? working, importance, and alternatives," *Available at https://www.spiceworks.com/tech/networking/articles/what-is-active-directory/*, 25 May 2023. [Online; accessed 31 July 2024].

[35] The CyberArk Corporation, "Understanding active directory components," *Available at https://subscription.packtpub.com/book/cloud-and-networking/9781801070393/1/ch01lvl1sec06/understanding-active-directory-components*, 2023. [Online; accessed 23-November-2023].

[36] Quest Corporation, "What are FSMO roles?," *Available at https://www.quest.com/learn/what-are-fsmo-roles.aspx*, 2023. [Online; accessed 24-November-2023].

[37] Microsoft, "Active directory domain services overview," *Available at https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview*, 2023. [Online; accessed 24-November-2023].

[38] B. Desmond, J. Richards, R. Allen, and A. G. Lowe-Norris, *Chapter 4 Naming Contexts and Application Partitions*. O'Reilly amp; Associates, 2013.

[39] J. M. Kizza, "Authentication," *Computer Network Security*, pp. 233–256, 2005.

[40] J. Kadlec, D. Jaros, and R. Kuchta, "Implementation of an advanced authentication method within microsoft active directory network services," in *2010 6th International Conference on Wireless and Mobile Communications*, pp. 453–456, IEEE, 2010.

[41] C. Metz, "Aaa protocols: authentication, authorization, and accounting for the internet," *IEEE Internet Computing*, vol. 3, no. 6, pp. 75–79, 1999.

[42] A. by Frontegg, "Authorization: A complete guide," *Available at https://frontegg.com/guides/authorization-a-complete-guide*, 07 November 2022. [Online; accessed 31 July 2024].

[43] The Microsoft Corporation, "Administrative tools and logon types," *Available at https://learn.microsoft.com/en-us/windows-server/identity/securing-privileged-access/reference-tools-logon-types*, 2023. [Online; accessed 14-December-2023].

[44] Authentication Protocols, "Security identifiers," *Available at https://www.blumira.com/authentication-protocols-101/*, 2023. [Online; accessed 14-December-2023].

[45] Microsoft, "Windows authentication architecture," *Available at https://learn.microsoft.com/en-us/windows-server/security/windows-authentication/windows-authentication-architecture*, 2023. [Online; accessed 17-December-2023].

[46] The Microsoft Corporation, "Security identifiers," *Available at https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-identifiers*, 2023. [Online; accessed 14-December-2023].

[47] The Microsoft Corporation, "Ntlm overview," *Available at https://learn.microsoft.com/en-us/windows-server/security/kerberos/ntlm-overview*, 2023. [Online; accessed 17-December-2023].

[48] Massachusetts Institute of Technology, "What is kerberos?," *Available at https://web.mit.edu/KERBEROS/#what_is*, 2023. [Online; accessed 17-December-2023].

[49] N. H. M. M. D. S. M. S. Sharon Goldberg, Miro Haller and A. Suhl, "Blast radius," *Available at https://www.blastradius.fail/*. [Online; accessed 07 August 2024].

[50] T. Grippo and H. A. Kholidy, "Detecting forged kerberos tickets in an active directory environment," *CoRR*, vol. abs/2301.00044, 2023.

[51] The MITRE Corporation, "Cve records," *Available at https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=kerberos*, 2023. [Online; accessed 30-December-2023].

[52] Tim MEDIN, Red Siege, "Attacking microsoft kerberos kicking the guard dog of hades," *Available at https://www.redsiege.com/wp-content/uploads/2020/08/Kerberoastv4.pdf*, 2014. [Online; accessed 06-January-2024].

[53] By QOMPLX, "Kerberoasting attacks explained," *Available at https://www.qomplx.com/blog/qomplx-knowledge-kerberoasting-attacks-explained/*, 6 May 2020. [Online; accessed 07-January-2024].

[54] Qomplx Knowledge, "Qomplx knowledge: Kerberoasting attacks explained," *Available at https://www.redsiege.com/wp-content/uploads/2020/08/Kerberoastv4.pdf*, 2023. [Online; accessed 06-January-2024].

[55] MIT, "Mit kerberos documentation," *Available at https://web.mit.edu/kerberos/krb5-1.13/doc/admin/pkinit.html?highlight=pkinit*, Release: 1.13.7. [Online; accessed 07 August 2024].

[56] L. Kotlaba, *Detection of Active Directory attacks*. PhD thesis, Bachelor's thesis. Czech Technical University in Prague, 2019.

[57] StrongDM Team, "Kerberoasting," *Available at https://www.strongdm.com/what-is/kerberoasting*. [Online; accessed 07-January-2024].

[58] H. S. Javitz, A. Valdes, *et al.*, "The sri ides statistical anomaly detector.," in *IEEE Symposium on Security and Privacy*, pp. 316–326, Oakland, 1991.

[59] Center for Internet Security, Inc., "Election security spotlight – signature-based vs anomaly-based detection," *Available at https://www.cisecurity.org/insights/spotlight/cybersecurity-spotlight-signature-based-vs-anomaly-based-detection*, 2024. [Online; accessed 16-January-2024].

[60] W. Zhang, Q. Yang, and Y. Geng, "A survey of anomaly detection methods in networks," in *2009 International Symposium on Computer Network and Multimedia Technology*, pp. 1–3, IEEE, 2009.

[61] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[62] Microsoft, "Protect your network," *Available at https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/network-protection?view=o365-worldwide*, 13 September 2023. [Online; accessed 13-January-2024].

[63] Sean Metcalf, "Detecting kerberoasting activity part 2 – creating a kerberoast service account honeypot," *Available at https://adsecurity.org/?p=3513*, 08 February 2017. [Online; accessed 13-January-2024].

[64] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: a statistical anomaly approach," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 76–82, 2002.

[65] M. All, "Supervised machine learning," *Available at https://www.datacamp.com/blog/supervised-machine-learning*, August 2022. [Online; accessed 31 July 2024].

[66] A. A. Ghorbani, W. Lu, M. Tavallaee, A. A. Ghorbani, W. Lu, and M. Tavallaee, "Theoretical foundation of detection," *Network Intrusion Detection and Prevention: Concepts and Techniques*, pp. 73–114, 2010.

[67] C. Zhang, G. Zhang, and S. Sun, "A mixed unsupervised clustering-based intrusion detection model," in *2009 Third International Conference on Genetic and Evolutionary Computing*, pp. 426–428, IEEE, 2009.

[68] S. Omar, A. Ngadi, and H. H. Jebur, "Machine learning techniques for anomaly detection: an overview," *International Journal of Computer Applications*, vol. 79, no. 2, 2013.

[69] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine learning*, vol. 109, no. 2, pp. 373–440, 2020.

[70] Michael Fuchs, "The data science process (crisp-dm)," *Available at https://michael-fuchs-python.netlify.app/2020/08/21/the-data-science-process-crisp-dm/*, 2020. [Online; accessed 30-April-2024].

[71] A. E. Mohamed, "Comparative study of four supervised machine learning techniques for classification," *International Journal of Applied*, vol. 7, no. 2, pp. 1–15, 2017.

[72] P. Mooijman, C. Catal, B. Tekinerdogan, A. Lommen, and M. Blokland, "The effects of data balancing approaches: A case study," *Applied Soft Computing*, vol. 132, p. 109853, 2023.

[73] P. Michaud and C. Persico, "Marshmallows kerberoasting," *Available at https://redcanary.com/blog/threat-detection/marshmallows-and-kerberoasting/*, Originally published May 11, 2022. Last modified April 30, 2024. [Online; accessed 30 July 2024].

[74] Microsoft, "Secure group managed service accounts," *Available at https://learn.microsoft.com/en-us/entra/architecture/service-accounts-group-managed*, 2022. [Online; accessed 18-May-2024].

[75] Rahul Bajaj, "Feature selection techniques in machine learning)," *Available at https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/*, 2024. [Online; accessed 04-May-2024].

[76] C. Pearson's, "Comparison of values of pearson's and spearman's correlation coefficients," *Comparison Of Values Of Pearson's And Spearman's Correlation Coefficients*, 2011.

[77] A. Gholamy, V. Kreinovich, and O. Kosheleva, "Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation," *Int. J. Intell. Technol. Appl. Stat*, vol. 11, no. 2, pp. 105–111, 2018.

[78] Data-Driven Science, "7 stages of machine learning — a framework," *Available at https://medium.com/@datadrivenscience/7-stages-of-machine-learning-a-framework*, 2020. [Online; accessed 04-May-2024].

[79] Splunk Cisco Company, "Getting started with machine learning at splunk," *Available at https://www.splunk.com/en*$_u s/blog/platform/getting - started - with - ml - at - splunk.html$, 2022. $[Online; accessed\ 18 - May - 2024]$.

[80] H. J. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," *arXiv preprint arXiv:2007.07588*, 2020.

[81] M. Alenazi and S. Mishra, "Cyberatttack detection and classification in iiot systems using xgboost and gaussian naïve bayes: A comparative study," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15074–15082, 2024.

[82] K. LTD, "Naive bayes classifiers: Types and use cases," *Available at https://keylabs.ai/blog/naive-bayes-classifiers-types-and-use-cases/*, 2024. [Online; accessed 27 October 2024].

[83] J. Markey and A. Atlasis, "Using decision tree analysis for intrusion detection: a how-to guide," *Sans Institute*, 2011.

[84] S. Singh, S. Agrawal, M. Rizvi, and R. S. Thakur, "Improved support vector machine for cyber attack detection," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 2011.

[85] A. Assiri, "Anomaly classification using genetic algorithm-based random forest model for network attack detection.," *Computers, Materials & Continua*, vol. 66, no. 1, 2021.

[86] S. S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective intrusion detection system using xgboost," *Information*, vol. 9, no. 7, p. 149, 2018.

[87] M. K. Islam, P. Hridi, M. S. Hossain, and H. S. Narman, "Network anomaly detection using lightgbm: A gradient boosting classifier," in *2020 30th international telecommunication networks and applications conference (ITNAC)*, pp. 1–7, IEEE, 2020.

[88] O. Nizan and A. Tal, "k-nnn: Nearest neighbors of neighbors for anomaly detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1005–1014, 2024.

[89] F. Amin and M. Mahmoud, "Confusion matrix in binary classification problems: a step-by-step tutorial," *Journal of Engineering Research*, vol. 6, no. 5, pp. 0–0, 2022.

[90] Natassha Selvaraj, "Confusion matrix, precision, and recall explained)," *Available at https://www.kdnuggets.com/2022/11/confusion-matrix-precision-recall-explained.html*, 2022. [Online; accessed 03-May-2024].

[91] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

APPENDIX A

CODES FOR DATA ANALYSIS

All python codes were shared on Github [1].

A.0.1 Correlation Heatmap

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Read input file
input_file = r"C:\Users\lenovo\Desktop\Dataset.csv"
df = pd.read_csv(input_file, delimiter=';')

# Step 2: Select only numeric columns
numeric_df = df.select_dtypes(include=[float, int])

# Check if numeric_df has any numeric columns
if numeric_df.empty:
    print("No numeric columns found in the dataset.")
else:
    # Step 3: Calculate correlation matrix
    corr_matrix = numeric_df.corr()

    if corr_matrix.empty:
        print("Correlation matrix is empty.")
    else:
        # Step 4: Visualize correlation matrix with Heatmap
        plt.figure(figsize=(10, 8))  # Adjust the figure size as needed
        sns.heatmap(corr_matrix, annot=True, cmap="coolwarm",
fmt='.2f', linewidths=0.5)
        plt.title('Correlation Matrix Heatmap')
        plt.show()
```

Figure 30: Code for Correlation Heatmap

[1] Phyton Codes: https://github.com/ysnakst/Dataset-for-Kerberoasting/blob/main

A.0.2 Explanatory Data Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Read CSV file
input_file = r"C:\Users\lenovo\Desktop\Dataset.csv"
df = pd.read_csv(input_file, delimiter=';')

# Statistical Informations
print(df.describe())

# Analysing Missing Values
print("\nMissing Values:")
missing_data = df.isnull().sum()
print(missing_data[missing_data > 0])

# Distribution and Visualization
plt.figure(figsize=(12, 6))
df.hist(bins=20, figsize=(20, 15), layout=(5, 4), color='blue')
plt.tight_layout()
plt.show()

# Outlier Analysis with Boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df)
plt.show()
```

Figure 31: Code For Explanatory Data Analysis

A.0.3 Feature Importance

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB, BernoulliNB
from sklearn.preprocessing import StandardScaler
from sklearn.inspection import permutation_importance
import xgboost as xgb
import lightgbm as lgb
import matplotlib.pyplot as plt
import numpy as np

# Split the features and target column
X = df.drop(columns=['issuspicous'])
y = df['issuspicous']

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.3, random_state=42)

# Initialize models including XGBoost and LightGBM
models = {
    'SVC': SVC(kernel='linear'),
    'RandomForest': RandomForestClassifier(),
    'KNN': KNeighborsClassifier(),
    'LogisticRegression': LogisticRegression(),
    'DecisionTree': DecisionTreeClassifier(),
    'GaussianNB': GaussianNB(),
    'BernoulliNB': BernoulliNB(),
    'XGBoost': xgb.XGBClassifier(use_label_encoder=False,
eval_metric='logloss'),
    'LightGBM': lgb.LGBMClassifier()
}

# Dictionary to store feature importances
feature_importances = {}
```

Figure 32: Feature Importance Code Part-1

```
# Calculate feature importance for each model
for model_name, model in models.items():
    model.fit(X_train, y_train)

    # If the model has feature_importances_ or coef_ attribute, use it
    if hasattr(model, 'feature_importances_'):
        feature_importances[model_name] = model.feature_importances_
    elif hasattr(model, 'coef_'):
        feature_importances[model_name] = np.abs(model.coef_).flatten()
    else:
        # For models without feature_importances_ or coef_, use
permutation importance
        result = permutation_importance(model, X_test, y_test,
n_repeats=10, random_state=42, n_jobs=-1)
        feature_importances[model_name] = result.importances_mean

# Plot feature importances for each model as line charts
plt.figure(figsize=(12, 8))
for model_name, importance in feature_importances.items():
    plt.plot(range(len(importance)), importance, label=model_name)

plt.xticks(np.arange(X.shape[1]), df.columns[:-1], rotation=90)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance Scores by Model')
plt.legend()
plt.tight_layout()
plt.show()
```

Figure 33: Feature Importance Code Part-2

APPENDIX B

CODES FOR MODEL EVALUATION

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
import xgboost as xgb
import lightgbm as lgb
from sklearn.naive_bayes import GaussianNB, BernoulliNB
from sklearn.metrics import confusion_matrix, precision_score,
recall_score, f1_score, fbeta_score, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Load the dataset for K-Fold Cross Validation
input_file = r"C:\Users\lenovo\Desktop\Dataset\2.Train-Validate.csv"
df = pd.read_csv(input_file, delimiter=';')

# Replace commas with dots
df = df.replace(',', '.', regex=True)

# Split the features and target column
X = df.drop(columns=['issuspicious'])
y = df['issuspicious']

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Define models and their hyperparameter grids
models = {
    'SVC': (SVC(), {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}),
    "GaussianNB": (GaussianNB(), {}),
    "BernoulliNB": (BernoulliNB(), {'alpha': [0.5, 1.0, 2.0]}),
```

Figure 34: Model Evaluation Code Part-1

```
    'RandomForest': (RandomForestClassifier(), {'n_estimators': [50,
100], 'max_depth': [10, 20]}),
    'KNN': (KNeighborsClassifier(), {'n_neighbors': [3, 5, 7]}),
    'LogisticRegression': (LogisticRegression(), {'C': [0.1, 1, 10]}),
    'DecisionTree': (DecisionTreeClassifier(), {'max_depth': [5, 10,
15]}),
    'XGBoost': (xgb.XGBClassifier(), {'learning_rate': [0.01, 0.1],
'max_depth': [3, 5]}),
    'LightGBM': (lgb.LGBMClassifier(), {'learning_rate': [0.01, 0.1],
'max_depth': [3, 5]}),
}

# Stratified K-Fold Cross Validation setup
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Dictionary to store the best models and their hyperparameters
best_params = {}

# GridSearch for each model
for model_name, (model, params) in models.items():
    grid_search = GridSearchCV(model, params, cv=skf, scoring='f1')
    grid_search.fit(X_scaled, y)
    best_params[model_name] = grid_search.best_params_

# Function to calculate performance metrics
def calculate_metrics(y_true, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    f2 = fbeta_score(y_true, y_pred, beta=2)
    accuracy = accuracy_score(y_true, y_pred)
    return tn, fp, fn, tp, precision, recall, f1, f2, accuracy

# Dictionary to store metrics for all models during Cross Validation
metrics = {}

for model_name, (model, params) in models.items():
    model_best = model.set_params(**best_params[model_name])
    cv_metrics = []

    for train_index, test_index in skf.split(X_scaled, y):
        X_train, X_test = X_scaled[train_index], X_scaled[test_index]
```

Figure 35: Model Evaluation Code Part-2

```
        y_train, y_test = y[train_index], y[test_index]
        model_best.fit(X_train, y_train)
        y_pred = model_best.predict(X_test)
        cv_metrics.append(calculate_metrics(y_test, y_pred))

    # Average over the cross-validation folds
    avg_metrics = np.mean(cv_metrics, axis=0)
    metrics[model_name] = avg_metrics

# Create a DataFrame for performance metrics
metrics_df = pd.DataFrame(metrics, index=['TN', 'FP', 'FN', 'TP',
'Precision', 'Recall', 'F1', 'F2', 'Accuracy']).T

# Sort the DataFrame by F2 score
metrics_df_sorted = metrics_df.sort_values(by='F2', ascending=False)

# Add ranking to model names
ranked_model_names = [f"{i+1}. {name}" for i, name in
enumerate(metrics_df_sorted.index)]
metrics_df_sorted.index = ranked_model_names

# Print the sorted metrics
print("Results for K-Fold Stratified Cross Validation (Average):")
print(metrics_df_sorted)

# Plot the metrics as a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(metrics_df_sorted[['Precision', 'Recall', 'F1', 'F2',
'Accuracy']], annot=True, cmap='Blues')
plt.title('Results for K-Fold Stratified Cross Validation (Average)')
plt.show()

# Test datasets with different ratios of malicious samples
test_datasets = {
    "0.012 Ratio of Malicious Samples":
r"C:\Users\lenovo\Desktop\Dataset\3a.Test.csv",
    "0.008 Ratio of Malicious Samples":
r"C:\Users\lenovo\Desktop\Dataset\3b.Test.csv",
    "0.004 Ratio of Malicious Samples":
r"C:\Users\lenovo\Desktop\Dataset\3c.Test.csv"
}

# Function to test models on unseen data
```

Figure 36: Model Evaluation Code Part-3

```
def test_on_unseen_data(test_file, model_best):
    # Load the test dataset
    df_test = pd.read_csv(test_file, delimiter=';')

    # Replace commas with dots
    df_test = df_test.replace(',', '.', regex=True)

    # Split features and target
    X_test = df_test.drop(columns=['issuspicious'])
    y_test = df_test['issuspicous']

    # Standardize the test data
    X_test_scaled = scaler.transform(X_test)

    # Predict on the test data
    y_pred = model_best.predict(X_test_scaled)

    # Calculate performance metrics
    return calculate_metrics(y_test, y_pred)

# Dictionary to store metrics for all test datasets
test_metrics = {}

# Test each model on each test dataset
for ratio, test_file in test_datasets.items():
    model_metrics = {}

    for model_name, model in models.items():
        # Use the best model parameters found during Cross Validation
        model_best = model[0].set_params(**best_params[model_name])
        model_metrics[model_name] = test_on_unseen_data(test_file,
model_best)

    # Store metrics for this ratio
    test_metrics[ratio] = model_metrics

# Display the metrics for each unseen dataset
for ratio, metrics in test_metrics.items():
    print(f"Results for {ratio}:")
    metrics_df = pd.DataFrame(metrics, index=['TN', 'FP', 'FN', 'TP',
'Precision', 'Recall', 'F1', 'F2', 'Accuracy']).T
    metrics_df_sorted = metrics_df.sort_values(by='F2',
ascending=False)
```

Figure 37: Model Evaluation Code Part-4

```
    print(metrics_df_sorted)

    # Plot the metrics as a heatmap
    plt.figure(figsize=(12, 8))
    sns.heatmap(metrics_df_sorted[['Precision', 'Recall', 'F1', 'F2',
'Accuracy']], annot=True, cmap='Blues')
    plt.title(f'Results for {ratio} (Unseen Data)')
    plt.show()
```

Figure 38: Model Evaluation Code Part-5