

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

2-1-2025

Lateral Movement Attacks Datasets: Benchmarking, Challenges, and Solutions

Anas Salah Salem Abdelhamid Mabrouk
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Mabrouk, Anas Salah Salem Abdelhamid, "Lateral Movement Attacks Datasets: Benchmarking, Challenges, and Solutions" (2025). *Electronic Theses and Dissertations*. 9655.
<https://scholar.uwindsor.ca/etd/9655>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Lateral Movement Attacks Datasets: Benchmarking, Challenges, and Solutions

By

Anas Mabrouk

A Thesis

Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2024

© 2024 Anas Mabrouk

Lateral Movement Attacks Datasets: Benchmarking, Challenges, and Solutions

by

Anas Mabrouk

APPROVED BY:

N. Zhang
Department of Electrical and Computer Engineering

S. Samet
School of Computer Science

S. Saad, Co-Advisor
School of Computer Science

M. Mamun, Co-Advisor
National Research Council of Canada

December 6, 2024

DECLARATION OF CO-AUTHORSHIP AND PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is the result of joint research, as follows:

Chapter 2 of the thesis was co-authored with Dr. Saad and Dr. Mamun. Chapter 3 was co-authored with B.Sc. Mohamed Hatem, Dr. Saad, and Dr. Mamun. In Chapter 2, Dr. Saad and I contributed equally to summarizing and analyzing the literature on lateral movement datasets. Furthermore, Dr. Saad, Dr. Mamun, and I collaboratively developed the proposed definition of lateral movement. In Chapter 3, the idea of creating a lateral movement dataset and identifying that gap in the existing literature was done by Dr. Saad. The framework's various components were created through the combined efforts and guidance of Dr. Saad, Dr. Mamun, Mohamed Hatem, and myself, each contributing to different aspects of the work. Dr. Saad and I designed the testbed infrastructure, and I was responsible for its implementation. I designed and implemented the Benign Data Engine with guidance from Dr. Saad and Dr. Mamun. The implementation and execution of lateral movement attacks were primarily carried out by Mohamed Hatem, with my assistance and oversight from Dr. Saad and Dr. Mamun. Dr. Mamun and I mainly designed the Labeling Engine, and I implemented it under the guidance of Dr. Saad.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes 2 original papers that have been previously published/submitted for publication in peer reviewed journals, as follows:

Thesis Chapter	Publication title/full citation	Publication Status
Chapter 2	A. Mabrouk and S. Saad, "Lateral Movement Datasets Analysis," In Press, 2024. School of Computer Science, University of Windsor, Windsor, Canada.	In Press
Chapter 3	A. Mabrouk, M. Hatem, S. Saad, and M. Mamun, "LMDG: A Framework for Lateral Movement Datasets Generation," In Press Manuscript, 2024. School of Computer Science, University of Windsor, Windsor, Canada, Faculty of Information Engineering and Technology, German University Cairo, Cairo, Egypt, National Research Council of Canada, New Brunswick, Canada.	In Press

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Advanced Persistent Threats (APTs) pose a significant cybersecurity risk by leveraging sophisticated techniques, with lateral movement (LM) playing a central role in these attacks. Lateral movement allows adversaries to navigate through compromised networks, escalating privileges, and gaining access to critical resources over extended periods. However, the detection of lateral movement has been hindered by a lack of comprehensive, high-quality datasets that accurately reflect the diverse and evolving tactics used in such attacks. Existing datasets suffer from several limitations, including a scarcity of lateral movement instances, outdated attack patterns, and insufficient diversity in techniques and attack paths, especially in cloud-based environments. Moreover, automatic labeling methods for dataset creation are often imprecise, complicating the training of effective detection models.

This work addresses these challenges by proposing a new benchmark dataset specifically tailored for lateral movement attacks. We conduct a comprehensive analysis of existing lateral movement attack datasets, highlighting gaps and providing insights into the strengths and weaknesses of current approaches. In response, we introduce the Lateral Movement Dataset Generator (LMDG), a framework designed to generate high-quality datasets for lateral movement and APT detection. The LMDG framework automates the generation of benign network traffic, simulates realistic attack scenarios, and incorporates an innovative labeling technique called process tree labeling, which improves the accuracy of automatic labeling compared to existing methods.

Our contributions offer significant advancements in the development of lateral movement detection systems. The new dataset provides a valuable resource for training and evaluating machine learning models, while the LMDG framework offers a reproducible toolset for generating datasets that accurately represent real-world attack behaviors. This work lays the foundation for future research into multi-stage APT detection, enabling the development of holistic systems that can better defend against the evolving landscape of sophisticated cyber threats.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors, Dr. Saad and Dr. Mamun, for their invaluable mentorship throughout my research. Their patience, guidance, and unwavering support have been instrumental in the completion of this thesis. I truly appreciate their expertise and encouragement, which have significantly contributed to my academic and professional growth.

TABLE OF CONTENTS

DECLARATION OF CO-AUTHORSHIP AND PREVIOUS PUBLICATION	iii
ABSTRACT	vi
ACKNOWLEDGEMENTS	vii
LIST OF TABLES	xi
LIST OF FIGURES	xii
1 Introduction	1
References	3
2 Lateral Movement Datasets Analysis	7
2.1 Introduction	7
2.2 Datasets Analysis	10
2.2.1 LANL Datasets (2015, 2018)	11
2.2.2 DARPA Transparent Computing Engagement 3 (DARPA 2018)	13
2.2.3 DARPA Transparent Computing Engagement 5 (DARPA 2019)	13
2.2.4 DARPA Operationally Transparent Cyber 2019 (OpTC) . . .	14
2.2.5 CERT Insider Threat Test Dataset (2020)	15
2.2.6 PicoDomain Dataset (2020)	15
2.2.7 DARPA Intrusion Detection Datasets (1998, 1999, 2000) . . .	16
2.2.8 NDSec-1 Dataset (2017)	17
2.2.9 Pivoting Detection Dataset (2017)	18
2.2.10 StreamSpot Dataset (2016)	21
2.2.11 ISCX Intrusion Detection Evaluation Dataset (2012)	21
2.2.12 DAPT (2020)	22
2.2.13 Unraveled (2023)	23
2.3 Conclusion	23
References	24
3 LMDG: A Framework for Lateral Movement Datasets Generation	27
3.1 Introduction	27
3.2 Challenges of Cybersecurity Datasets Creation	30
3.2.1 General Challenges	30
3.2.2 Realistic Datasets	31
3.2.3 Synthetic Datasets	32
3.2.4 Semi-synthetic Datasets	32
3.3 LMDG Framework	33
3.3.1 Overview	33
3.3.2 Testbed Infrastructure	35

3.3.3	Benign Data Engine (BDE)	37
3.3.3.1	Sessions Scheduler	38
3.3.3.2	Sessions Executor	43
3.3.4	Attack Engine (AE)	47
3.3.4.1	Lateral Movement Attacks	47
3.3.4.2	Challenges in Automating Lateral Movement Attacks	50
3.3.4.3	A Candidate Solution	51
3.3.4.4	CALDERA as an Attack Engine	53
3.3.4.5	Lateral Movement Attacks in LMDG Dataset	56
3.3.4.5.1	<i>Second Attack Scenario</i>	57
3.3.4.5.2	<i>Third Attack Scenario</i>	57
3.3.4.5.3	<i>Fourth Attack Scenario</i>	59
3.3.4.5.4	<i>fifth Attack Scenario</i>	62
3.3.4.5.5	<i>sixth Attack Scenario</i>	63
3.3.4.5.6	<i>seventh Attack Scenario</i>	65
3.3.5	LMDG Labelling Engine (LE)	67
3.3.5.1	Challenges in Attack Data Labeling	67
3.3.5.2	LMDG Labeling Engine	68
3.3.5.2.1	<i>LMDG Labeling Engine Input</i>	69
3.3.5.2.2	<i>Attack Steps Forest Construction</i>	71
3.3.5.2.3	<i>System Logs Labeling</i>	73
3.3.5.2.4	<i>Network Traffic Labeling.</i>	74
3.4	Dataset	76
3.5	Qualitative Analysis	80
3.6	Related Work	82
3.6.1	Cybersecurity Datasets Generation	83
3.6.1.1	Frameworks	84
3.6.1.1.1	<i>AIT Framework</i>	84
3.6.1.1.2	<i>LADEMU Framework</i>	87
3.6.1.1.3	<i>CREME Framework</i>	90
3.6.2	Available Lateral Movement Datasets	92
3.6.2.0.1	<i>LANL Dataset 2015</i>	93
3.6.2.0.2	<i>LANL Dataset 2015</i>	96
3.6.2.0.3	<i>DARPA Transparent Computing En-</i> <i>gagement 5 (DARPA 2019)</i>	99
3.6.2.0.4	<i>DARPA Operationally Transparent Cy-</i> <i>ber 2019 (OpTC)</i>	100
3.6.2.0.5	<i>PicoDomain 2020</i>	100
3.6.2.0.6	<i>Pivoting Detection Dataset 2017</i>	101
3.6.2.0.7	<i>DAPT 2020</i>	102
3.6.2.0.8	<i>Unraveled Dataset 2023</i>	102
3.7	Discussion	105
3.8	Conclusions and Future Work	106
	References	108

4	Conclusion	118
	VITA AUCTORIS	120

LIST OF TABLES

3.4.1 Dataset Statistics	80
3.6.1 Examples of Malicious Authentications in LANL 2015 Dataset	96
3.6.2 Examples of Malicious Authentications in LANL 2015 Dataset	99

LIST OF FIGURES

2.1.1 A Lateral Movement Example.	10
2.2.1 LANL 2015 malicious authentications as directed graph.	12
2.2.2 Lateral Movement in PicoDomain Dataset. [8]	16
2.2.3 Lateral Movements at Pivoting Dataset at time t_1	19
2.2.4 Lateral Movements at Pivoting Dataset at time t_2	19
2.2.5 Lateral Movements at Pivoting Dataset at time t_3	20
2.2.6 Lateral Movements at Pivoting Dataset at time t_4	20
3.3.1 The network topology used to generate LMDG dataset.	36
3.3.2 Benign Data Engine (BDE) overview.	38
3.3.3 Frequency distribution of $t_{\text{start_abnormal_early}}$ from 03:30 AM to 07:29 AM over 20,000 trials. The distribution follows an exponential distribution with a rate parameter $\lambda = 0.00037$, indicating higher frequencies of abnormal early start times occurring at earlier minutes and tapering off towards later minutes.	44
3.3.4 Frequency distribution of $t_{\text{start_late}}$ from 08:31 AM to 10:00 AM over 20,000 trials. The distribution follows a flipped exponential distribu- tion with a rate parameter $\lambda = 0.0006$, demonstrating lower frequencies at earlier times and gradually increasing towards later times.	45
3.3.5 Frequency distribution of every minute from 07:30 AM to 08:30 AM of $t_{\text{start_on_time}}$ over 20,000 trials drawn by the Sessions Scheduler based on a normal distribution with confidence interval of 2.58.	45
3.3.6 Frequency distribution of every minute from 03:30 AM to 04:00 PM of t_{start} over 20,000 trials drawn by the Sessions Scheduler.	46

3.3.7	First Attack Scenario in LMDG dataset which is a Passing the Hash attack (PtH). This figure also illustrates the traversal behavior (hops) in the Passing the TGT attack scenario outlined in Subsection 3.3.4.5.1, providing a step-by-step visualization of the movement through network nodes.	49
3.3.8	Example command for obtaining an elevated shell using the pass-the-hash (PtH) technique via <i>Mimikatz</i>	51
3.3.9	Partial process tree illustrating the execution of the Pass-the-Hash (PtH) attack on DC2, as discussed in subsection 3.3.4.1.	52
3.3.10	Caldera Adversary representing attack step 3 from figure 3.3.7 . Each row corresponds to an ability, detailing the command name and associated MITRE ATT&CK tactics and techniques.	55
3.3.11	Third Attack Scenario in LMDG dataset which is AS-REP Roasting attack.	59
3.3.12	Fourth Attack Scenario in LMDG dataset which is an Advanced Delegation attack.	61
3.3.13	Fifth Attack Scenario in LMDG dataset which is a Password Spray attack.	63
3.3.14	Sixth Attack Scenario in LMDG dataset which is a Silver Ticket attack. This figure also illustrates the traversal behavior (hops) in the Golden Ticket attack scenario outlined in Subsection 3.3.4.5.6, providing a step-by-step visualization of the movement through network nodes. . .	65
3.3.15	Output of algorithm 3.3.2 showing two process trees rooted at the same malicious process p_r . The first tree, representing attack step 3, and the second tree, representing attack step 4 in Figure 3.3.7 explained in subsection 3.3.4.1.	73

3.4.1 Daily Distribution of Attack Steps: This histogram visualizes the frequency of attack steps executed over time, with each bar representing the count of attack steps occurring on a specific day. The x-axis denotes individual days, while the y-axis represents the number of occurrences.	78
3.4.2 Timeline of Attack Step Occurrences by Scenario: This scatter plot illustrates the timing of attack steps across various days, with each point representing the occurrence of an attack step on a specific day and time. The x-axis indicates the occurrence dates, while the y-axis represents the time of day to highlight daily distribution patterns. Each scenario is color-coded with a distinct hue, allowing for quick differentiation of scenarios.	79
3.4.3 Frequency Distribution of Scenario and Version pairs: This bar plot displays the count of occurrences for each distinct (Scenario, Version) pair. The x-axis represents individual combinations of scenarios and their respective versions. The y-axis shows the count of occurrences.	80
3.6.1 Abstracting the problem of cybersecurity datasets generation [51]. . .	84
3.6.2 labelling methodology in AIT.	85
3.6.3 LANL 2015 malicious authentications as directed graph.	95
3.6.4 LANL 2015 malicious authentications as directed graph.	98

CHAPTER 1

Introduction

In the ever-evolving landscape of cybersecurity, the threat landscape is characterized by increasingly sophisticated and persistent attacks. Advanced Persistent Threats (APTs) are among the most formidable, representing a class of cyberattacks that employ stealthy and prolonged strategies to infiltrate, control, and exfiltrate sensitive information from targeted networks. These attacks often exploit vulnerabilities to gain an initial foothold and, through a series of meticulous and interconnected steps, expand control within the compromised environment. Lateral movement, a pivotal tactic within APT campaigns, allows attackers to pivot across networked systems, escalating privileges and accessing critical resources that were initially out of reach. As cyber adversaries continue to refine their techniques, lateral movement has emerged as a key strategy for evading detection, maintaining access, and progressing toward their ultimate goals [1, 9, 19, 17].

The concept of lateral movement has been explored extensively in cybersecurity literature, with various definitions emphasizing its importance in the progression of cyberattacks. As defined by the MITRE ATT&CK framework, lateral movement involves the use of techniques that enable adversaries to enter and control remote systems within a network, with the end goal of gaining further access and discovering valuable targets [17]. However, the literature reveals a significant gap in the clarity and precision of this definition. While the general concept is widely acknowledged, there is a lack of a comprehensive and universally accepted framework to characterize lateral movement in its entirety. Existing definitions often provide a broad overview,

but they fail to capture the nuances and complexities that define this critical phase of an attack [12, 2, 15, 18]. This thesis aims to address this gap by providing a clear, precise, and comprehensive definition of lateral movement that can serve as the foundation for more effective detection models and strategies.

The challenge in detecting lateral movement is compounded by several factors. These attacks are often prolonged, with threat actors moving through networks over extended periods while blending in with normal network activity. Additionally, the sheer volume of data generated by enterprise networks makes it difficult to identify malicious activity amidst the noise. Attackers frequently exploit legitimate authentication credentials and system tools, further obscuring their actions. The complexity of detecting lateral movement is also heightened by the use of novel malware variants, zero-day exploits, and evasion techniques that allow attackers to bypass conventional detection mechanisms [5, 11, 7, 4, 3]. The growing sophistication of lateral movement tactics has made it a critical focus for cybersecurity research, with numerous efforts aimed at developing models for its early detection and mitigation [6].

Despite the importance of lateral movement detection, current research is hindered by challenges related to data quality. The effectiveness of machine learning (ML) models for detecting lateral movement depends heavily on the quality and accuracy of the datasets used for training and evaluation. Many existing datasets suffer from issues such as noisy labels, class imbalances, and insufficient diversity in attack patterns, limiting their usefulness for developing robust detection models [8, 13]. Furthermore, most datasets lack sufficient instances of lateral movement attacks, making it difficult to train models that can generalize across a wide range of attack scenarios [20, 10]. This thesis contributes to the field by addressing these challenges through the introduction of the Lateral Movement Datasets Generator (LMDG) framework, a comprehensive solution designed to generate high-quality lateral movement datasets. By automating the generation of benign and attack data, as well as the labeling process, the LMDG framework enables the creation of datasets that more accurately reflect real-world attack scenarios [10, 14].

The contributions of this thesis are threefold. First, we conduct a thorough anal-

ysis of existing cybersecurity benchmark datasets to assess their effectiveness in representing lateral movement attacks. Second, we introduce a novel lateral movement dataset that addresses many of the quality issues observed in current datasets, providing a valuable resource for training and evaluating detection models. Finally, we propose a new automatic labeling technique, process tree labeling, which offers a more accurate and scalable solution for labeling lateral movement activities in system and network logs. By providing a comprehensive framework for dataset generation, this research aims to advance the state of the art in lateral movement detection and improve the effectiveness of cybersecurity defense mechanisms against APTs [10, 16].

References

- [1] Adel Alshamrani et al. “A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2019), pp. 1851–1877. DOI: 10.1109/COMST.2019.2891891.
- [2] Giovanni Apruzzese et al. “Detection and Threat Prioritization of Pivoting Attacks in Large Networks”. In: *IEEE Transactions on Emerging Topics in Computing* 8.2 (2020), pp. 404–415. DOI: 10.1109/TETC.2017.2764885.
- [3] Giovanni Apruzzese et al. “Detection and Threat Prioritization of Pivoting Attacks in Large Networks”. In: *IEEE Transactions on Emerging Topics in Computing* 8 (2020), pp. 404–415. URL: <https://api.semanticscholar.org/CorpusID:64482369>.
- [4] Tim Bai et al. “RDP-based Lateral Movement detection using Machine Learning”. In: *Computer Communications* 165 (2021), pp. 9–19. ISSN: 0140-3664. DOI: <https://doi.org/10.1016/j.comcom.2020.10.013>. URL: <https://www.sciencedirect.com/science/article/pii/S0140366420319617>.

- [5] Haibo Bian et al. “Uncovering Lateral Movement Using Authentication Logs”. In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 1049–1063. DOI: 10.1109/TNSM.2021.3054356.
- [6] Atul Bohara et al. “An Unsupervised Multi-Detector Approach for Identifying Malicious Lateral Movement”. In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. 2017, pp. 224–233. DOI: 10.1109/SRDS.2017.31.
- [7] Benjamin Bowman et al. “Detecting Lateral Movement in Enterprise Computer Networks with Unsupervised Graph AI”. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. San Sebastian: USENIX Association, Oct. 2020, pp. 257–268. ISBN: 978-1-939133-18-2. URL: <https://www.usenix.org/conference/raid2020/presentation/bowman>.
- [8] Remi Denton et al. “Bringing the people back in: Contesting benchmark machine learning datasets”. In: *arXiv preprint arXiv:2007.07399* (2020).
- [9] Chenquan Gan et al. “Advanced Persistent Threats and Their Defense Methods in Industrial Internet of Things: A Survey”. In: *Mathematics* 11.14 (2023). ISSN: 2227-7390. DOI: 10.3390/math11143115. URL: <https://www.mdpi.com/2227-7390/11/14/3115>.
- [10] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. “Datasets are not enough: Challenges in labeling network traffic”. In: *Computers & Security* 120 (2022), p. 102810. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2022.102810>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404822002048>.
- [11] Xueyuan Han et al. “Unicorn: Runtime Provenance-Based Detector for Advanced Persistent Threats”. In: *Proceedings 2020 Network and Distributed System Security Symposium*. NDSS 2020. Internet Society, 2020. DOI: 10.14722/ndss.2020.24046. URL: <http://dx.doi.org/10.14722/ndss.2020.24046>.
- [12] Grant Ho et al. “Hopper: Modeling and Detecting Lateral Movement”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Associ-

- ation, Aug. 2021, pp. 3093–3110. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/ho>.
- [13] A. Kenyon, L. Deka, and D. Elizondo. “Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets”. In: *Computers & Security* 99 (2020), p. 102022. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2020.102022>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404820302959>.
 - [14] Meejoung Kim and Inkyu Lee. “Human-guided auto-labeling for network traffic data: The GELM approach”. In: *Neural networks* 152 (2022), pp. 510–526.
 - [15] Qingyun Liu et al. “Latte: Large-Scale Lateral Movement Detection”. In: *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. 2018, pp. 1–6. DOI: 10.1109/MILCOM.2018.8599748.
 - [16] Mark Mazumder et al. “DataPerf: Benchmarks for Data-Centric AI Development”. In: *ArXiv abs/2207.10062* (2022). URL: <https://api.semanticscholar.org/CorpusID:250699092>.
 - [17] *MITRE ATT&CK*[®]. 2024. URL: <https://attack.mitre.org/>.
 - [18] Emilie Purvine, John R. Johnson, and Chaomei Lo. “A Graph-Based Impact Metric for Mitigating Lateral Movement Cyber Attacks”. In: *Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense*. SafeConfig ’16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 45–52. ISBN: 9781450345668. DOI: 10.1145/2994475.2994476. URL: <https://doi.org/10.1145/2994475.2994476>.
 - [19] Branka Stojanović, Katharina Hofer-Schmitz, and Ulrike Kleb. “APT datasets and attack modeling for automated detection methods: A review”. In: *Computers & Security* 92 (2020), p. 101734. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2020.101734>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404820300213>.

- [20] Ngan Tran et al. “Data Curation and Quality Evaluation for Machine Learning-Based Cyber Intrusion Detection”. In: *IEEE Access* 10 (2022), pp. 121900–121923. DOI: 10.1109/ACCESS.2022.3211313.

CHAPTER 2

Lateral Movement Datasets

Analysis

ANAS MABROUK, SHERIF SAAD, AND MOHAMMAD MAMUN

In Press

2.1 Introduction

In Cybersecurity, the persistent evolution of cyber threats poses an ongoing challenge for organizations and individuals. Among the array of sophisticated tactics employed by threat actors, the concept of "lateral movement" has emerged as a pivotal strategy for adversaries seeking to maneuver within compromised network environments. As elucidated by the exposition in [12], Lateral Movement embodies an array of methodologies engaged by malevolent entities to infiltrate and orchestrate control over remote network systems. The attainment of their intended goals is frequently characterized by the imperative act of pivoting across an assortment of interconnected systems and accounts. Corresponding definitions mirroring this conception of Lateral Movement are also extant within the literature, as expounded upon in [5], [1], [10], and [15], delineating the concept as the orchestrated movement of an attacker from a primary host to successive nodes within a compromised network, culminating in the pursuit of a designated target.

Our investigation into lateral movement detection through a comprehensive literature review has revealed a lack of a precise and comprehensive definition for this concept. As an illustration, consider the definition provided by MITRE ATT&CK

[12], which characterizes lateral movement as follows: "Lateral Movement consists of techniques that adversaries use to enter and control remote systems on a network. Following through on their primary objective often requires exploring the network to find their target and subsequently gaining access to it. Reaching their objective often involves pivoting through multiple systems and accounts." This definition, while informative, offers an overview of the lateral movement concept without a profound, detailed explanation. Similarly, other literature sources showed comparable definitions as mentioned above. Cybersecurity literature offers a wide array of models designed for detecting lateral movement, APTs, anomalies, and threat hunting. It is a reasonable expectation for these models to identify instances of lateral movement whenever they occur. However, it's essential to recognize that we cannot detect or effectively combat something that remains vague and unclear. As such, it becomes crucial to establish a precise and clear definition of lateral movement. This precision is necessary for developing models that can accurately detect it and for evaluating the effectiveness of existing models in identifying this particular threat.

Let's begin by establishing two fundamental concepts: horizontal progression and vertical progression. Horizontal progression entails obtaining an initial foothold on one or multiple hosts within a network, with each initial access executed independently of the others. For instance, consider a scenario where a network scan reveals ten hosts within a segment. Among these, three have distinct vulnerabilities that can be exploited for access. It's important to note that each of these initial accesses occurs in isolation. This form of horizontal progression, while significant for gaining initial access, does not qualify as lateral movement. On the contrary, vertical progression involves accessing multiple systems where these accesses are interdependent. To illustrate, an adversary might secure an initial foothold within network segment A, proceed to segment B, and then advance to segment C. Importantly, these initial accesses are not isolated but instead rely on one another. For instance, the adversary gains control over a host in segment A, providing remote access to a machine in segment B. From a machine in segment B, further access is obtained to a machine

in segment C. ***Lateral movement***, therefore, can be defined as the vertical progression between hosts, accounts, or the transition from one set of privileges to another. Lateral movement across hosts occurs when an adversary gains the ability to access host B from host A. Hosts A and B may exist within the same subnet or across different network segments. Lateral movement between accounts, on the other hand, transpires when an adversary secures access to account B after having initially accessed account A. Account B may either possess higher privileges than account A or provide access to specific resources that account A lacks access to. Additionally, we can extend our perspective to encompass situations where an adversary, through privilege escalation, attains access to a specific set of resources. This can be regarded as a form of lateral movement, as the adversary transitions from one set of resources (privileges) to another.

An illustrative instance of lateral movement, as per the defined concept, is presented in Figure 2.1.1 in the form of a directed graph. In this scenario, an adversary initially secures access to host H1 using account A1, along with a set of privileges denoted as P1. The adversary then undertakes privilege escalation, transitioning to a distinct privilege set, P2. Subsequently, the intruder accesses a new host, H3, employing a fresh account, A7, and privileges designated as P5. Finally, access is extended to host H5 using the same account and associated privilege set. This example demonstrates three intermediate steps or "hops" between distinct states or nodes within the graph. It's important to note that each atomic lateral movement represents a single hop within such a graph, which can be aggregated to form more intricate pivoting behaviors. Consequently, lateral movement scenarios can be deconstructed into these atomic movements, representing the fundamental building blocks of lateral movement.

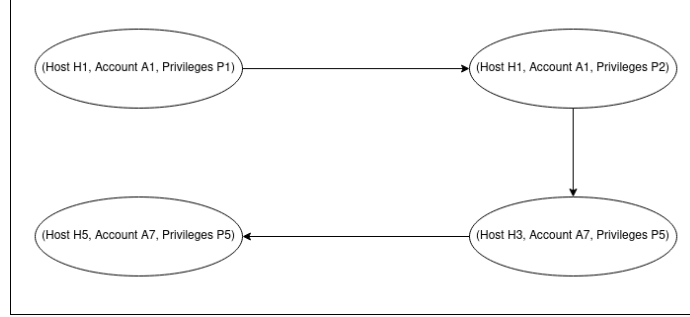


Fig. 2.1.1: A Lateral Movement Example.

When discussing the cloud environment, we can apply a similar concept with a slight adjustment. In the cloud, we encounter Identities and permissions (or policies), which can be likened to the Accounts and Privileges mentioned in our previous definition. Identities encompass user accounts, as well as application and service accounts, as they require authentication, similar to users when accessing resources. Permissions or policies, on the other hand, correspond to the privileges in our earlier definition, essentially outlining the permissions associated with each identity.

One key distinction is that, instead of having hosts or resources directly, there exists an additional layer known as the services layer. These services, like AWS EC2 or S3 buckets, offer resources such as compute instances and object storage. Consequently, in the context of lateral movement in the cloud, we observe a vertical progression encompassing identities, permissions/policies, services, and resources. A recent example highlighted by the Microsoft Threat Intelligence team [6] involved adversaries gaining initial access to an Azure-based database server through SQL injection. Subsequently, they attempted to obtain a cloud identity token using the IMDS (Instance Metadata Service) to access other cloud resources.

2.2 Datasets Analysis

Commencing with our analysis, we will examine the extant open-source datasets that have been employed within the scholarly discourse encompassing the domains of Lateral Movement detection, Advanced Persistent Threat (APT) detection, Intrusion

Detection, and Threat hunting. The objective of this scrutiny is to ascertain the presence of instances pertaining to lateral movement within these datasets.

2.2.1 LANL Datasets (2015, 2018)

Two datasets originating from Los Alamos National Laboratory’s corporate (LANL), namely, the ”Unified Host and Network Data Set” [17] and the ”Comprehensive, Multi-Source Cyber-Security Events” (LANL 2015) [7]. The LANL 2018 dataset constitutes a subset of network and host events procured from the LANL enterprise network during an approximately 90-day timeframe; notably, this dataset does not encompass any annotated instances of malicious events, thereby precluding its utility in the evaluation of models for Lateral Movement detection. Conversely, the LANL 2015 dataset comprises a collection of Windows-based authentication events originating from individual computing nodes and centralized Active Directory domain controller servers spanning a 58-day duration. Additionally, it encapsulates process initiation and termination events sourced from individual Windows-based machines, Domain Name Service (DNS) query activities as observed on internal DNS servers, network flow data originating from various key router locations, and an explicitly delineated array of red teaming exercises designed to exemplify malicious authentication behaviors.

Upon meticulous examination of the malicious authentication incidents, it becomes evident that the manifestation of lateral movement is absent, substantiated by the absence of the pivotal traversal between disparate hosts, as stipulated by the aforementioned definitional parameters. To expound further, a directed graph can be meticulously crafted to depict the interplay of these authentication activities among hosts. An anticipated outcome within this contextualized representation would be the emergence of pathways embodying a compositional magnitude of two or beyond. Regrettably, the empirical observation reveals that the most protracted trajectory within this directed graph remains confined to a singular step. The graphical illus-

tration of the entire gamut of red teaming authentication instances is encompassed within Figure 3.6.4.

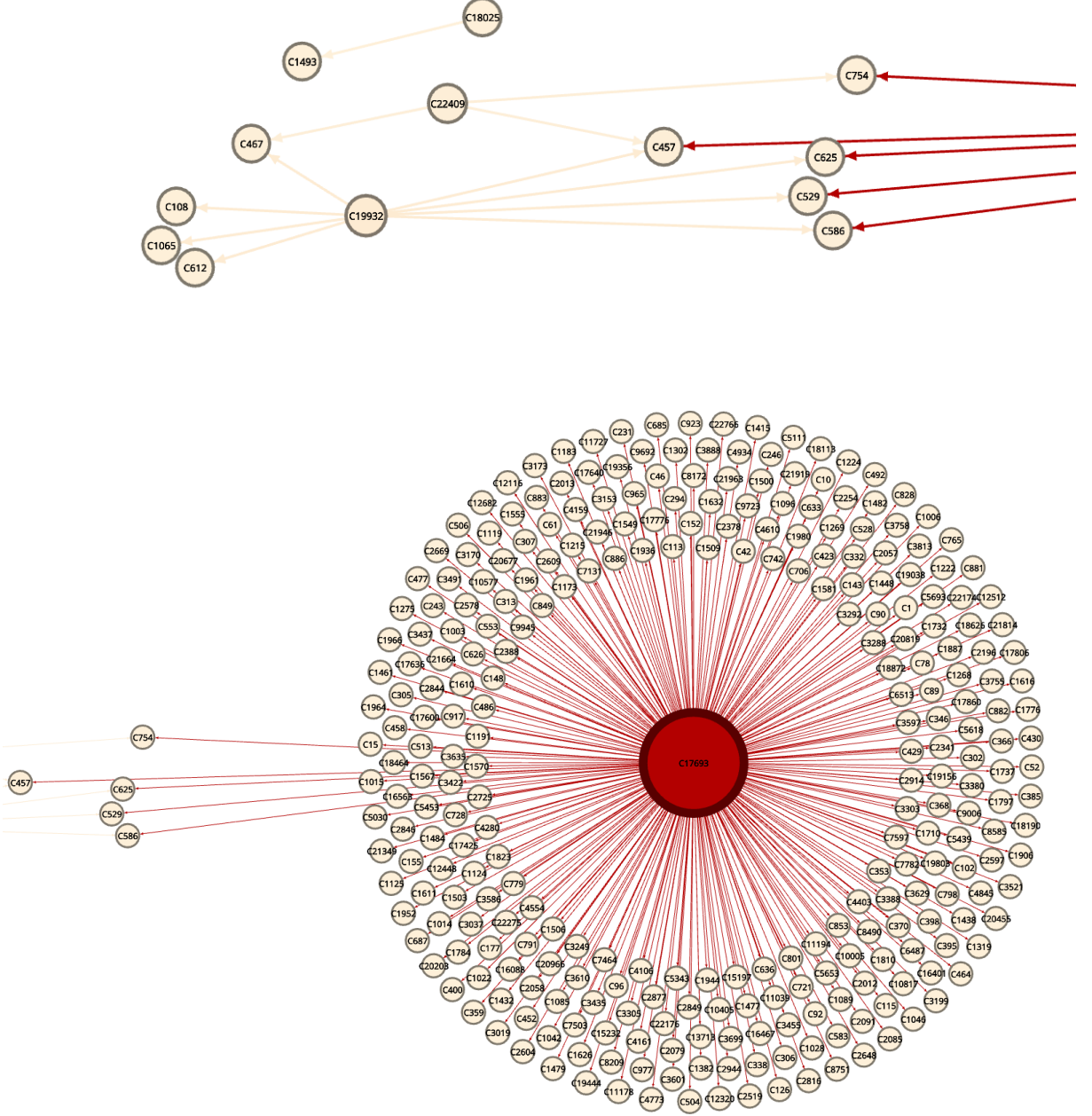


Fig. 2.2.1: LANL 2015 malicious authentications as directed graph.

2.2.2 DARPA Transparent Computing Engagement 3 (DARPA 2018)

In this engagement’s scope, six hosts running diverse platforms, notably Nginx on FreeBSD, Ubuntu 14.04, Ubuntu 12.04, Windows 10, and Android 6.0.1. The commencement of the engagement transpired with an initial phase characterized by the generation of benign data, where a meticulously scripted sequence of operations was executed on each host. Following the benign data generation phase, control over the testing environment was relinquished to the offensive team, who subsequently initiated a sequence of maneuvers aimed at emulating the behaviors exhibited by both novel and pre-existing Advanced Persistent Threats (APTs) across the entirety of the test range. Throughout this period, a continuous stream of benign background traffic was sustained. At the same time, activities of a malicious nature were exclusively carried out between 9 am and 5 pm on weekdays, and the engagement lasted for five days. Upon examining the attack scenarios within this dataset, it becomes evident that various tactics have been employed. These encompass reconnaissance, privilege escalation, command-and-control communication, and data exfiltration. Notably, the dataset does not manifest any instances of lateral movement.

2.2.3 DARPA Transparent Computing Engagement 5 (DARPA 2019)

The configuration within this dataset closely resembled that of Engagement 3, albeit encompassing a larger group of hosts. The assembly consisted of 16 distinct hosts that operated on diverse operating systems, namely Windows, Ubuntu, and Android, mirroring the compositional framework of the preceding dataset. Before and during the engagement, There was a phase of benign data generation. All instances of attack materialized exclusively between 9 a.m. and 5 p.m. on weekdays across eight days. In contradistinction to the third engagement, the present one comprises two lateral movement scenarios. The first scenario is characterized by a sequence wherein attackers successfully compromise a host within the targeted network, configuring it

to function as their command and control hub. After this, they pivoted to another Linux-based host using stolen authentication credentials. The second scenario closely parallels the first one, involving a similar strategy wherein attackers initially gained a foothold on the network, subsequently pivoting onto multiple intermediary hosts through SSH and stolen credentials. Within this dataset, these two instances are the exclusive manifestations of lateral movement. These instances diverge from the protracted temporal characteristics commonly associated with lateral movements as they transpire over a short interval. Both instances share a commonality in their approach, employing an identical technique to accomplish lateral maneuvering.

2.2.4 DARPA Operationally Transparent Cyber 2019 (OpTC)

Compared to Engagement 3 and 5, this dataset has many hosts, a thousand hosts in a Windows network, and the data from five hundred hosts was collected rather than from the complete set of hosts due to space constraints. The evaluation started with benign record generation, followed by the red team attacks, which were performed in three days. Benign traffic ran continuously during red team activity. Kafka, an open-source stream-processing server, facilitates information sharing among system components. Windows 10 endpoints employ sensors to monitor host events, packaging them into JSON records sent to Kafka. These records are then translated into eCAR format by a server and reinserted into Kafka. A data analytics component further processes the eCAR records, converting them into a graph structure for analysis and visualization. Within this dataset, two occurrences of lateral movement are identifiable. The initial incident occurred on the first day, involving a sequence of four intermediary transitions across five distinct hosts, with one of these hosts designated the domain controller. The attacker employed Windows Management Instrumentation (WMI) to effectuate the traversal between hosts, augmenting the process by integrating additional techniques. The subsequent occurrence unfolded the next day, likewise leveraging WMI; however, it exhibited greater complexity than its predecessor, characterized by a larger number of intermediary transitions. Similar to the circumstances in Engagement 5, this dataset exhibits a limited number of instances of

lateral movement occurring within a concise timeframe, underscored by a deficiency in the array of strategies employed for accomplishment.

2.2.5 CERT Insider Threat Test Dataset (2020)

The CERT Insider Threat Dataset [9] is a valuable resource in cybersecurity, offering a comprehensive collection of real-world instances and data related to insider threats. Insider threats, which involve malicious or negligent actions taken by individuals within an organization, pose a substantial risk to data security, intellectual property, and operational integrity. The CERT Insider Threat Dataset significantly enhances our ability to identify, prevent, and mitigate insider-driven security breaches by providing access to a diverse array of documented cases and associated data points. The dataset contains five scenarios of insider threats. An example for these scenarios is as follows: an individual previously unaccustomed to utilizing removable storage devices or engaging in post-business hours activities initiates a pattern of logging into the system after operational hours, employing removable storage media for data interaction, and subsequently transmitting data to the domain "wikileaks.org." Following these actions, this individual promptly disengages from organizational affiliations. This dataset does not contain any instances of lateral movement [12], as stated in the malicious scenarios description.

2.2.6 PicoDomain Dataset (2020)

The PicoDomain [8] simulation comprised a compact Windows office setting encompassing five workstations, a domain controller, and a gateway firewall/router. This setup is connected to a limited-scale internet housing websites and adversary infrastructure. The internal network featured a Windows Active Directory environment with distinct Organizational Units (OUs): HR, R&D, and a confidential supersecret OU. Scripts mimicking web browsing and SMB file sharing were utilized, and data collection spanned three days. The Mandiant Attack Lifecycle (MAL) [3] was the framework for outlining the adversary's campaign strategy, mainly focusing on the

recurring phases of hostile campaigns. The initial MAL stages include Initial Reconnaissance, Initial Compromise, and Establish Foothold. Subsequent phases, repeated as necessary, encompass Escalate Privileges, Internal Reconnaissance, Move Laterally, and Maintain Presence. The MAL concludes with the adversary accomplishing their mission. The dataset reveals prevalent attacker engagement in multiple stages. Notably, lateral movement (LM) predominantly utilized WMI and DCOM techniques with minimal diversity. The dataset comprises a single LM scenario, shown in figure 2.2.2, over a short three-day span, lacking instances with an extended temporal scope.

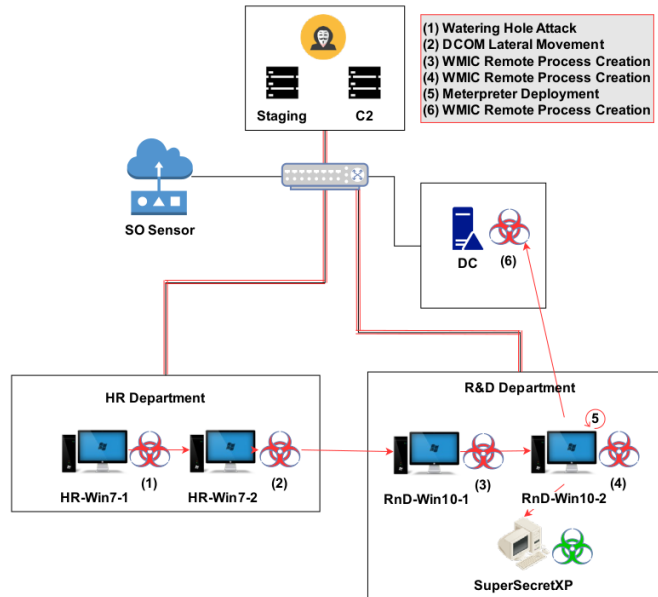


Fig. 2.2.2: Lateral Movement in PicoDomain Dataset. [8]

2.2.7 DARPA Intrusion Detection Datasets (1998, 1999, 2000)

MIT Lincoln Laboratory has generated a six-week training dataset along with two-weeks testing dataset for the 1998 DARPA Intrusion Detection Evaluation [4]. The training data spans six weeks, with the initial two weeks seeing gradual additions of background traffic and attacks, reaching a steady state. The subsequent four weeks remain consistent in terms of attack types and background traffic. Data collection occurs from 8 AM to 6 AM the next day on weekdays, giving 22 hours of daily data. This involves continuous data collection on a network simulating over 1000

virtual hosts and 700 users, split into inside and outside segments. There are 2 routers, 2 hubs, fewer than 40 inside hosts, and approximately 13 outside hosts. After a thorough examination of the dataset, we’ve observed the absence of Lateral Movement instances. In total, the dataset encompasses more than 100 instances of 20 distinct attack types, encompassing activities such as denial of service, remote unauthorized access, local privilege escalation, surveillance, probing, and anomalous user behavior. It’s noteworthy to point out that there is a single type of attack labeled ”multihop,” which, despite its name, could be argued as not fitting the lateral movement category since it lacks the pivotal traversal between disparate hosts, as defined in the introduction.

2.2.8 NDSec-1 Dataset (2017)

The authors created the dataset [2] by setting up a network with two parts: a private network (used by a company or organization) and a simulated Internet. These parts were divided by a router that acted as a NAT gateway and firewall for the private network. The private side had various workstations and machines running different Windows and Linux versions. A tcpdump sensor captured traffic within the private network. Log event data were collected on each host. The dataset contains three attack scenarios, in which we can consider the first two as demonstration of lateral movement. In the first attack, a machine simulated a compromised BYOD host within a secure network. Attack methods such as brute-forcing, ARP, and DNS spoofing were employed against SSH, email, and web servers. The second scenario involved a brute-force attack on a web server, resulting in an SQL injection that accessed login details and password hashes from the database. This impacted a small user group, where customized malware was utilized to encrypt files on two hosts and report to an external server. It’s noticeable that this dataset comprises just two scenarios, both executed within a single day. The lateral movement path’s extent can be seen as encompassing two hops.

2.2.9 Pivoting Detection Dataset (2017)

Within this dataset [1], network traffic information takes shape as network flows observed within a large organization throughout a day. These flows embody the interactions of internal hosts within the observed network setting. Each flow sample in the dataset carries a binary label, indicating its involvement in a pivoting activity. The labeling process underwent manual execution and validation. The dataset’s size reaches about 6 GB, encompassing close to 75 million network flows. The dataset predominantly records pivoting activities where the ”pivoter” host controlled the ”terminal” host remotely using third-party tools like the Windows Remote Desktop protocol. It’s important to note that these actions are typical, non-malicious pivoting activities that occurred within the monitored organization. As such, they depict infrequent and harmless occurrences. As previously noted, this dataset was gathered within a brief timeframe of just one day. This limited duration may not be an ideal representation of Lateral Movement, which often occurs over an extended period. Upon creating a graph that visualizes all the pivoting activities with their temporal progression within the dataset in figures 2.2.3 2.2.4 2.2.5 2.2.6, it becomes evident that there are only a few instances of Lateral Movement, all of which involve two hops. The specific technique employed in these pivoting instances remains unclear; however, it is noteworthy that all of them were executed over TCP.

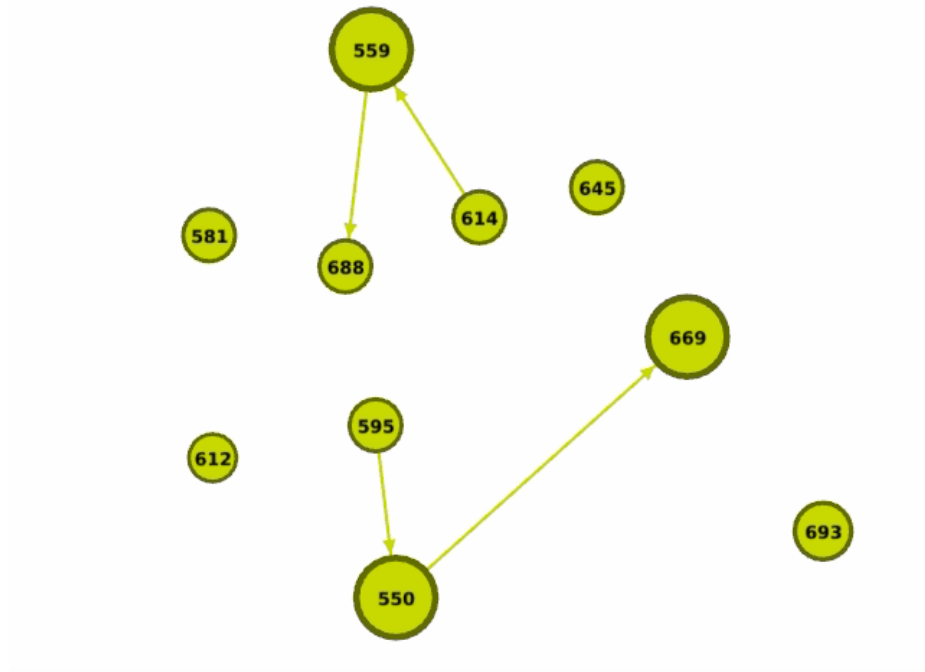


Fig. 2.2.3: Lateral Movements at Pivoting Dataset at time t_1

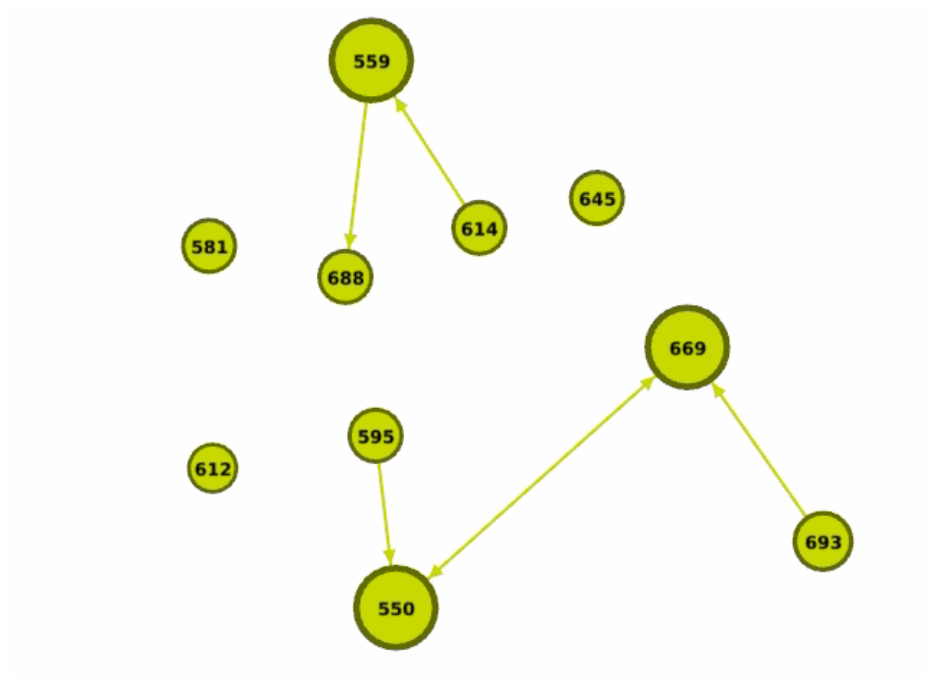


Fig. 2.2.4: Lateral Movements at Pivoting Dataset at time t_2

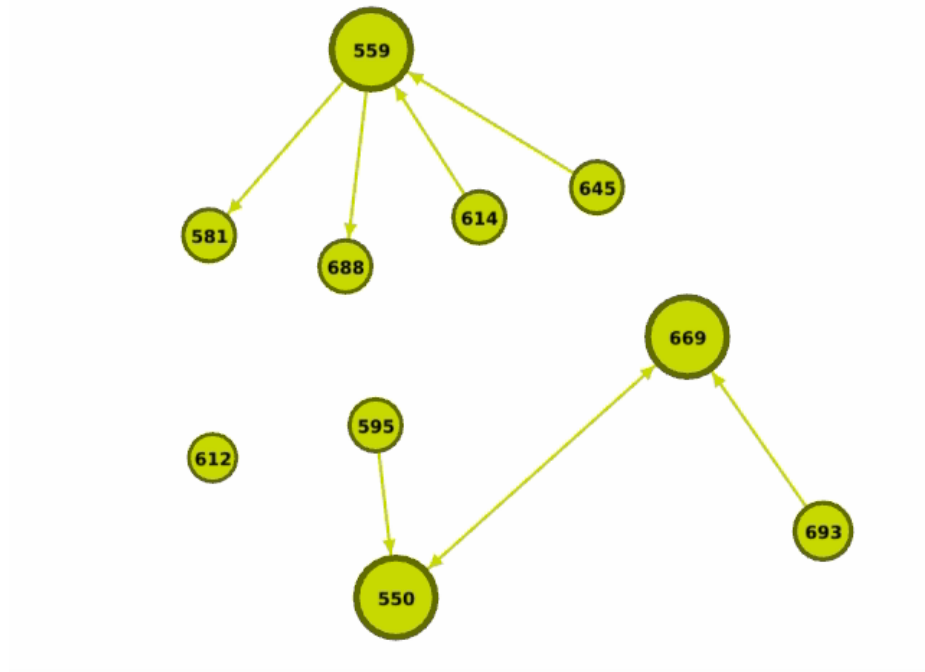


Fig. 2.2.5: Lateral Movements at Pivoting Dataset at time t_3

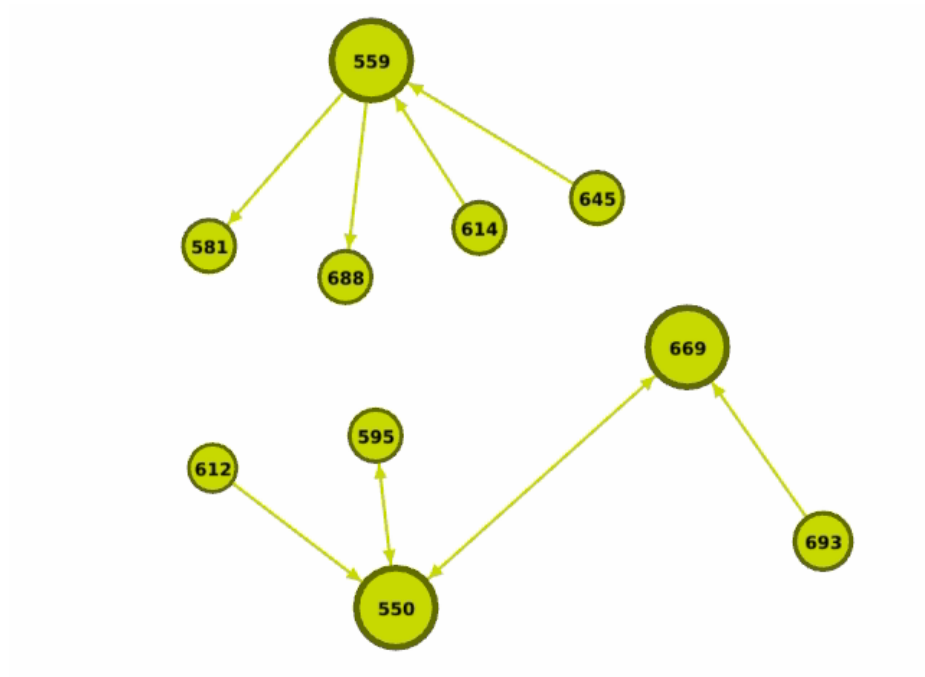


Fig. 2.2.6: Lateral Movements at Pivoting Dataset at time t_4

2.2.10 StreamSpot Dataset (2016)

The creation of this dataset [11] aimed to address the challenge of identifying anomalies within a continuous flow of heterogeneous graphs that consist of various node and edge types. The goal was to achieve real-time detection using limited memory resources. This problem finds its roots in security applications, particularly in detecting advanced persistent threats (APTs) at the host level. The dataset comprises flow-graphs originating from one attack and five benign scenarios, all executed on a single host. The benign scenarios encompass routine online activities such as browsing YouTube, downloading files, visiting cnn.com, using Gmail, and playing a video game. The attack scenario involves a drive-by download through a malicious URL, exploiting a Flash vulnerability to gain root access on the host. To capture these scenarios, Selenium Remote Control automated the execution of 100 tasks. The flow graphs for each task were constructed using traced system calls from the task’s initiation until completion. As explained this dataset does not contain any instances of lateral movement.

2.2.11 ISCX Intrusion Detection Evaluation Dataset (2012)

The experimental network setup in [16] comprises 21 interconnected Windows workstations that are operating on different versions of the Windows OS, namely Windows XP and 7. These workstations have been distributed among four separate LANs, while a fifth LAN has been designated for servers offering essential services such as web hosting, email communication, DNS (Domain Name System), and Network Address Translation (NAT). The NAT server has a dual role: it serves as the gateway for the network’s connection to the Internet. It concurrently functions as a firewall, permitting legitimate communication and obstructing unauthorized access attempts. Positioned as the network’s primary hub, the main server undertakes responsibilities, including hosting the network’s website, managing email services, and serving as the internal name resolver. Additionally, a secondary server is dedicated to handling internal ASP.NET applications. The main server and the NAT server run a Linux

operating system. The paper’s description of the attack scenarios reveals four distinct attacks spanning a week. Upon analyzing these attacks, it becomes apparent that each case involved a series of lateral movements, encompassing two successive hops, passing through three hosts. The sequence of lateral movements consistently began with the targeted hosts receiving a deceptive email containing a malicious PDF file harboring a reverse TCP shell. Subsequently, both the initial and secondary hops followed a uniform approach, exploiting hosts operating Windows XP and utilizing a vulnerable SMB authentication protocol. Notably, there was only a single occurrence where the second hop was executed through a brute force method to ascertain user credentials. It is evident from the dataset that there needs to be more varied methodologies for executing lateral movement. Additionally, all instances of lateral movement follow a consistent pattern, consisting of a predetermined sequence spanning two hops. These movements occur briefly, confined explicitly to a single day.

2.2.12 DAPT (2020)

The DAPT2020 [13] dataset was developed as an Advanced Persistent Threat (APT) dataset with two primary objectives: to make attacks indistinguishable from normal traffic and to include traffic across both the public-to-private interface and the internal (private) network. The testbed used was minimal, consisting of two virtual machines—one connected to a private network and the other to a public network—along with a log server and a gateway router. This simplified architecture represents a limitation, as it does not accurately reflect the complexity of real-world environments. Data collection spanned five days, with the first day capturing only normal traffic and the subsequent four days containing various stages of an APT attack. On the fourth day, the lateral movement phase occurred, involving reconnaissance and exploitation activities from the compromised public VM to gain access to critical systems on the internal network. This phase employed tools and techniques such as Nmap for network scanning, the vsftpd 2.3.4 vulnerability, weak SSH authentication, a MySQL script for CVE-2012-2122, and Metasploit. However, the dataset includes only a lateral movement instance executed over a 10-hour window—a significantly shorter

timeframe than realistic LM attacks, which can span weeks or even months.

2.2.13 Unraveled (2023)

The Unraveled dataset [14] builds upon the DAPT2020 dataset, introducing significant enhancements. The testbed architecture has been substantially improved, emulating a realistic enterprise network environment. The system architecture separates corporate and production networks with a firewall. The organization has 15 employees, using Snort as a Network Intrusion Detection System (NIDS) monitored by a Blue Team. The corporate network contains three subnets, each simulating a department with different operating systems. Logs are sent to a centralized ELK server in the production network. The production network consists of a public subnet with a web server and a honeypot and a private subnet hosting critical services like a database, internal application, and mail server. A firewall regulates traffic, allowing only specific public-to-private connections, while private servers can access the public network freely. An additional enhancement in the Unraveled dataset was the extended execution of the APT attack over six weeks. However, the lateral movement phase remained relatively simplistic, occurring within a single day and consisting of internal reconnaissance and password cracking. In both the DAPT2020 and Unraveled datasets, the attack execution and labeling processes were conducted manually.

2.3 Conclusion

In summary, the current datasets face significant challenges, highlighting the need to develop a new dataset that effectively addresses these issues. These challenges encompass a shortage of Lateral Movement instances in existing datasets, which hinders model training and generalization. Furthermore, the limited diversity in Lateral Movement techniques and the often short timeframes associated with these activities present additional obstacles in creating robust detection models. Moreover, the

prevalence of Lateral Movement paths comprising only a few hops limits the current datasets’ scope and symbolic value. The absence of dedicated datasets tailored for Cloud-based Lateral Movement scenarios further underscores the need for comprehensive and up-to-date resources in this domain. Another critical concern is the obsolescence of existing datasets, rendering them inadequate for capturing recent attack patterns and trends. Lastly, some datasets offer only a partial view of the overall threat landscape, focusing solely on network flow data, thus emphasizing the necessity for more comprehensive datasets encompassing a broader spectrum of Lateral Movement activities. Addressing these challenges in dataset creation is paramount to advancing the efficacy of Lateral Movement detection models in cybersecurity.

References

- [1] Giovanni Apruzzese et al. “Detection and Threat Prioritization of Pivoting Attacks in Large Networks”. In: *IEEE Transactions on Emerging Topics in Computing* 8.2 (2020), pp. 404–415. DOI: 10.1109/TETC.2017.2764885.
- [2] Frank Beer et al. “A new Attack Composition for Network Security”. In: *10. DFN-Forum Kommunikationstechnologien*. Bonn: Gesellschaft für Informatik e.V., 2017, pp. 11–20. ISBN: 978-3-88579-665-7.
- [3] Mandiant (A FireEye Company). “Targeted Attack Lifecycle”. In: (2023). URL: <https://www.mandiant.com/resources/insights/targeted-attack-lifecycle>.
- [4] “DARPA Intrusion Detection Evaluation Data Set”. In: (1998). URL: <https://archive.ll.mit.edu/ideval/data/1998data.html>.
- [5] Grant Ho et al. “Hopper: Modeling and Detecting Lateral Movement”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3093–3110. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/ho>.

- [6] Microsoft Threat Intelligence. “Defending new vectors: Threat actors attempt SQL Server to cloud lateral movement”. In: (October 3, 2023). URL: <https://www.microsoft.com/en-us/security/blog/2023/10/03/defending-new-vectors-threat-actors-attempt-sql-server-to-cloud-lateral-movement/>.
- [7] Alexander D. Kent. “Cybersecurity Data Sources for Dynamic Network Research”. In: *Dynamic Networks in Cybersecurity*. Imperial College Press, June 2015.
- [8] Craig Laprade, Benjamin Bowman, and H. Howie Huang. “PicoDomain: A Compact High-Fidelity Cybersecurity Dataset”. In: *CoRR* abs/2008.09192 (2020). arXiv: 2008.09192. URL: <https://arxiv.org/abs/2008.09192>.
- [9] Brian Lindauer. “Insider Threat Test Dataset”. In: (Sept. 2020). DOI: 10.1184/R1/12841247.v1. URL: https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247.
- [10] Qingyun Liu et al. “Latte: Large-Scale Lateral Movement Detection”. In: *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. 2018, pp. 1–6. DOI: 10.1109/MILCOM.2018.8599748.
- [11] Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. “Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs (KDD16)”. In: ().
- [12] *MITRE ATT&CK*[®]. 2024. URL: <https://attack.mitre.org/>.
- [13] Sowmya Myneni et al. “DAPT 2020-constructing a benchmark dataset for advanced persistent threats”. In: *Deployable Machine Learning for Security Defense: First International Workshop, MLHat 2020, San Diego, CA, USA, August 24, 2020, Proceedings 1*. Springer. 2020, pp. 138–163.
- [14] Sowmya Myneni et al. “Unraveled—A semi-synthetic dataset for Advanced Persistent Threats”. In: *Computer Networks* 227 (2023), p. 109688.

- [15] Emilie Purvine, John R. Johnson, and Chaomei Lo. “A Graph-Based Impact Metric for Mitigating Lateral Movement Cyber Attacks”. In: *Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense*. SafeConfig '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 45–52. ISBN: 9781450345668. DOI: 10.1145/2994475.2994476. URL: <https://doi.org/10.1145/2994475.2994476>.
- [16] Ali Shiravi et al. “Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection”. In: *Comput. Secur.* 31.3 (May 2012), pp. 357–374. ISSN: 0167-4048. DOI: 10.1016/j.cose.2011.12.012. URL: <https://doi.org/10.1016/j.cose.2011.12.012>.
- [17] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. “Unified Host and Network Data Set”. In: *Data Science for Cyber-Security*. World Scientific, Nov. 2018. Chap. Chapter 1, pp. 1–22. DOI: 10.1142/9781786345646_001. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9781786345646_001. URL: https://www.worldscientific.com/doi/abs/10.1142/9781786345646_001.

CHAPTER 3

LMDG: A Framework for Lateral Movement Datasets Generation

ANAS MABROUK, MOHAMED HATEM, SHERIF SAAD, AND MOHAMMAD MAMUN

In Press

3.1 Introduction

Advanced Persistent Threats (APTs) represent a sophisticated category of cyberattacks characterized by their prolonged and stealthy presence within a targeted computer system or network, aimed at ultimately exfiltrating sensitive data or causing significant harm [1, 31, 76]. APTs employ a diverse array of techniques and tactics meticulously crafted to circumvent the defensive mechanisms of the victim’s security infrastructure [59].

Among the array of sophisticated techniques employed by advanced threat actors, the concept of "Lateral Movement" has emerged as a critical strategy for adversaries seeking to maneuver within compromised network environments. As elucidated by the exposition in [59], Lateral Movement embodies an array of methodologies engaged by malevolent entities to infiltrate and orchestrate control over remote network systems. The attainment of their intended goals is frequently characterized by the imperative act of pivoting across an assortment of interconnected systems and accounts. Corresponding definitions mirroring this conception of Lateral Movement are also extant within the literature, as expounded upon in [41], [3], [54], and [65], delineating the concept as the orchestrated movement of an attacker from a primary host to succes-

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

sive nodes within a compromised network, culminating in the pursuit of a valuable target.

Lateral movement-based attacks are becoming a growing threat to large private and government networks, frequently causing information exfiltration and service disruptions [12]. Analyzing various APT campaigns reveals that nearly all employ lateral movement to navigate networks. The purpose of lateral movement is to transition from one system to another, infiltrating additional resources and gaining higher privileges. This process enables attackers to discover and collect valuable data, expand their control over the targeted organization, and maintain long-term access to the compromised IT infrastructure [81, 1, 31, 71]. Since lateral movement is a crucial phase in an APT attack, early detection is vital to minimize losses and prevent attackers from gaining further access to the network [10].

Detecting Lateral Movement attacks poses a significant challenge, primarily due to several factors; firstly, the prolonged duration of these attacks, which can extend over months, significantly complicates their detection. Additionally, the sheer volume of enterprise traffic provides adversaries ample opportunities to blend in and seamlessly remain undetected amidst regular network activity. Various tactics and techniques exist for executing Lateral Movement attacks, often leaving traces within network and system logs [59]. Attackers can effectively evade detection mechanisms by leveraging legitimate authentication credentials, system tools, and other evasion techniques. Furthermore, the prevalence of false security alerts further adds to the difficulty of distinguishing genuine threats from benign anomalies. Moreover, the incorporation of zero-day exploits or novel malware variants as part of these attacks further amplifies the complexity of detection [10, 37, 14, 6, 4].

Current research endeavors for lateral movement detection rely on machine learning [74, 12, 53, 77, 55]. The machine learning paradigm depends heavily on datasets to train and evaluate detection models, and the quality of these datasets directly impacts model performance and evaluation accuracy. Without high-quality training data, models can exhibit performance discrepancies, reducing accuracy and increasing false positives [25, 45] (see section 3.2). A growing body of literature explores

the evidence supporting that neglecting the fundamental importance of data has led to inaccuracies and bias in ML models [57]. For instance, researchers in [16] demonstrated that even minor modifications to a benchmark dataset significantly impact model performance more than the specific machine learning technique. Therefore, better data quality is essential to improve generalization and avoid bias in machine learning models [5, 69].

Most cybersecurity datasets suffer from quality issues, particularly those containing lateral movement attacks. Common data quality problems include noisy labels, insufficient labeling, class imbalance, limited diversity of attack patterns, outdated attack types, simplistic synthetic generation environments, and short generation periods (see section 3.6). Additionally many existing datasets either lack instances of lateral movement attacks altogether or contain only a limited number of such instances [79, 45, 35, 17]. Consequently, developing a comprehensive dataset, or ideally a framework, that addresses these challenges and others is essential for advancing research in lateral movement detection.

To this end, our paper introduces a framework called LMDG (Lateral Movement Datasets Generator), which addresses most of the issues discussed in sections 3.2 and 3.6. Our contributions can be summarized as follows:

- Conducting a thorough analysis of current cybersecurity benchmark datasets to assess the presence of lateral movement attacks. For datasets containing LM attacks, we analyze the properties of these attacks, including the number of LM attacks, diversity of techniques used, the time frame of the attacks, number of hops or LM movements, data sources collected from these attacks (e.g., authentication logs, network flows), labeling methods employed, and testbed architecture used 3.6. To the best of our knowledge, this study presents the first analysis specifically tailored for evaluating lateral movement datasets.
- Creating a benchmark dataset focused on lateral movement attacks that address many of the existing issues in current LM datasets and conducting a qualitative

analysis of it 3.4 3.5. This dataset will be valuable for the research community in training and evaluating LM detection models.

- Developing the LMDG framework used to generate the dataset, providing a reproducible framework for creating lateral movement/APT datasets (see section 3.3). The framework automates the generation of benign data (i.e., normal employees behavior) 3.3.3, the execution of attack scenarios 3.3.4, and, crucially, the labeling process (i.e., labeling the records in the system and network logs associated with the attacks). Automatic labeling is particularly challenging in lateral movement datasets due to benign internal hosts performing malicious activities. In the literature on cybersecurity dataset generation, three primary automatic labeling techniques are discussed: Injection Timing, Behavior Profiles, and Network Security Tools [35, 47]. We propose a new automatic labeling technique, *process tree labeling*, which we argue is better and more accurate than all other automatic labeling techniques 3.3.5.

3.2 Challenges of Cybersecurity Datasets Creation

As indicated by [45, 75, 78], the lack of high-quality public datasets significantly hinders the experimentation and evaluation of Intrusion Detection Systems (IDS), especially anomaly-based detectors. This scarcity arises from several challenges, which can be categorized into four groups: general challenges and those specific to realistic, synthetic, and semi-synthetic datasets.

3.2.1 General Challenges

Complex attacks, e.g., Advanced Persistent Threats (APTs), do not follow a uniform path and continually evolve, exploiting new vulnerabilities and tools to stay effective and are performed over prolonged periods. Hence, datasets need to cover extensive periods, often several months, which presents significant challenges in data storage

and processing [76]. Moreover, accurately labeling such large datasets requires expert knowledge, adding to the burden [76]. Despite these challenges, more research is needed to develop consistent metrics to assess the realism of datasets in terms of live network performance and the distribution of anomalies and threats [45]. Furthermore, data quality issues such as inconsistency, duplication, incompleteness, lack of comprehension, absence of variety, and imprecise timestamps can compromise the effectiveness of machine learning models [79]. The difficulty in obtaining representative and accurately labeled datasets is compounded by the rapid evolution of malicious behaviors, which quickly renders existing datasets obsolete [35].

3.2.2 Realistic Datasets

Experimentation and data collection on live networks are typically infeasible, especially for systems with business or mission-critical functions or where the data contains sensitive information [45]. In addition, specialized malware generators and well-insulated network infrastructure are required to limit the damage from any malware or threat simulations [45]. The primary challenge arises from the sensitive nature of the data: inspecting network traffic can reveal highly sensitive information, including confidential or personal communications, an organization’s business secrets, or its users’ network access patterns. Any breach of such information can be catastrophic for the organization and affected third parties, leading researchers to face insurmountable organizational and legal barriers when attempting to provide datasets to the community [75]. Organizations capable of producing and publishing representative and accurate data are often reluctant due to the risk of exposing sensitive information, while efforts to anonymize data are considered prohibitively costly [35]. Although sanitizing captured data by removing or anonymizing sensitive information has been tried, these efforts have seen limited success due to the persistent fear that information can still leak, a well justified concern; additionally, maintaining such datasets can be prohibitively expensive [75].

3.2.3 Synthetic Datasets

Creating representative environments for IDS datasets requires a skillful design that includes numerous network assets such as simulated servers, clients, routers, and switches, making these datasets challenging to create and maintain [45]. Existing datasets often represent only a synthetic subset of infrastructure, and network representation is a critical issue identified in analyzing several major public labeled datasets [45]. Due to the scarcity of public data, researchers frequently need to assemble their datasets. However, this is challenging as most researchers lack access to appropriately sized networks. The activity in a small laboratory network fundamentally differs from the aggregate traffic seen in larger networks where NIDSs are deployed, making it difficult to generalize conclusions from small environments to more extensive settings [75, 17]. While synthetic datasets can be free of sensitivity concerns, realistically, simulating Internet traffic is difficult. Anomaly detection systems evaluated using only simulated activity often lack realism and relevance [75, 17]. Additionally, synthetic datasets have several drawbacks, including the absence of noise, leading to unrealistically good detection results, a limited range of threat event types, a lack of label accuracy, and potential biases from following an insufficiently accurate synthetic user model [76, 45]. The absence of a systematic approach for dataset generation that can be frequently updated further complicates the creation of high-quality datasets [35, 17].

3.2.4 Semi-synthetic Datasets

Semi-synthetic datasets combine realistic data, often collected from network traffic or system logs, with synthetic data. This fusion offers advantages but also inherits challenges from both sources. As noted in [73], semi-synthetic datasets can suffer from limitations in both real and synthetic data. Integration between the two data types can be complex, ensuring they reflect real-world relationships. Additionally, the synthetic component can introduce bias if not carefully constructed [76]. This necessitates techniques that ensure the synthetic data accurately reflects the statistical

properties of the real data, preventing the model from learning unrealistic patterns.

3.3 LMDG Framework

3.3.1 Overview

The LMDG framework leverages virtualization technologies, specifically using VirtualBox, to simulate organizational networks. As discussed in Section 3.2, synthetically generated datasets offer several advantages, such as the ability to create controlled and repeatable experimental conditions and the flexibility to simulate a wide range of attack scenarios and network configurations. However, they also come with limitations, such as a potential lack of realism and the challenge of accurately mimicking real-world traffic patterns and user behaviors. The LMDG framework addresses these downsides by incorporating advanced virtualization techniques and realistic scenario generation, thereby enhancing the fidelity and utility of the synthetic datasets for research in lateral movement detection. One of the key features of VirtualBox is its ability to configure virtual machines to join various types of virtual networks, including NAT networks, Bridged networks, Internal networks, and Host-Only networks, among others. This capability provides significant flexibility in simulating diverse organizational network environments. Each type of virtual network serves distinct purposes: NAT networks facilitate internet access for virtual machines while isolating them from the host network; Bridged networks allow virtual machines to appear as separate entities on the physical network, enabling direct communication with other physical and virtual devices; Internal networks restrict communication to virtual machines within the same network, enhancing security and isolation for specific test environments; and Host-Only networks enable communication solely between the virtual machines and the host system, without external network access. These diverse networking options empower researchers to accurately emulate the unique topologies and structures of different organizational networks, thereby enhancing the realism and applicability of the simulations for research in lateral movement detection and

other cybersecurity studies.

Active Directory (AD), Microsoft’s directory service, is widely recognized as the most popular solution for managing and organizing IT profiles within organizations, facilitating essential functions such as authentication, authorization, and accounting. Its extensive adoption stems from its robust capabilities in centralized user and resource management, policy enforcement, and security administration across Windows-based environments [61, 58]. The recent CrowdStrike incident underscored the profound dependence of businesses and organizations on Windows-based systems. The large-scale outage disrupted critical IT operations across numerous industries, halting business activities and leading to substantial financial losses [64, 46].

Building on the virtualization technology, the LMDG framework leverages Windows Domains and Active Directory to simulate comprehensive network software configurations within organizational settings. This integration enables the creation of realistic and dynamic network environments that closely mimic real-world organizational infrastructures, enhancing the accuracy and relevance of the generated datasets for research in lateral movement detection and other cybersecurity applications. By utilizing these technologies, the LMDG framework provides a valuable tool for studying and mitigating advanced persistent threats (APTs) and other complex cyber-attacks in environments that reflect the actual operational scenarios of many enterprises.

For log collection, the LMDG framework employs packet-capturing technology, such as Wireshark, to acquire network traffic traces that can subsequently be processed to create network flow records. This method ensures detailed and granular capture of network interactions, providing a rich dataset for analysis. A continuously running service is deployed on every host and gateway within the simulated environment to ensure robustness. This service is designed to capture and store network traffic persistently, even in the event of system crashes, thereby providing the integrity and continuity of the traffic data. Due to its comprehensive and robust logging capabilities, the LMDG framework leverages Windows Event Logs regarding system log collection. Windows Event Logs offer detailed system and application event records,

including security events, system performance data, and operational diagnostics. This extensive logging is critical for creating accurate and reliable datasets, as it captures various events and activities within the Windows environment. By integrating these technologies, the LMDG framework ensures the collection of high-fidelity data essential for developing and evaluating advanced cybersecurity detection models, particularly in the context of lateral movement and other sophisticated attack vectors.

In the following subsections, we will provide a detailed discussion of our network topology 3.3.2, which can be easily extended and tailored to various topologies based on the target enterprise network. We will also elaborate on the Benign Data Engine (BDE) 3.3.3, crucial in generating realistic benign user behavior essential for any credible dataset. Furthermore, we will describe the Attack Engine 3.3.4, which automates adversary emulation to facilitate flexible and efficient execution of attacks. Finally, we will focus on the most significant and innovative component of the LMDG framework, the Labelling Engine (LE) 3.3.5. The LE can automatically and accurately extract records associated with attacks from network and system logs with minimal noise, utilizing *process tree labeling* method. This capability is particularly challenging yet essential, as internal hosts often execute lateral movement attacks.

3.3.2 Testbed Infrastructure

This network, showed in figure 3.3.1, simulates a small-sized company with five departments, each residing in a distinct network segment with its dedicated Windows domain. For instance, the Sales department operates within the domain of *sales.lmt.com* and is situated in the subnet *192.168.59.0/24* with its dedicated domain controller *DC 3*. Three additional subnets are present in the network configuration: one signifies the root Windows domain *lmt.com*, another accommodates the company’s servers, and a third denotes a DMZ, i.e., *192.168.0.0/24* which is part of the IT Windows domain. Routers facilitate connections between these diverse subnets. Naturally, the structure of this network can be adjusted and expanded as needed.

In our experimental setup, VirtualBox networking was utilized to configure network segmentation. All subnets were established as internal networks, isolating them

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

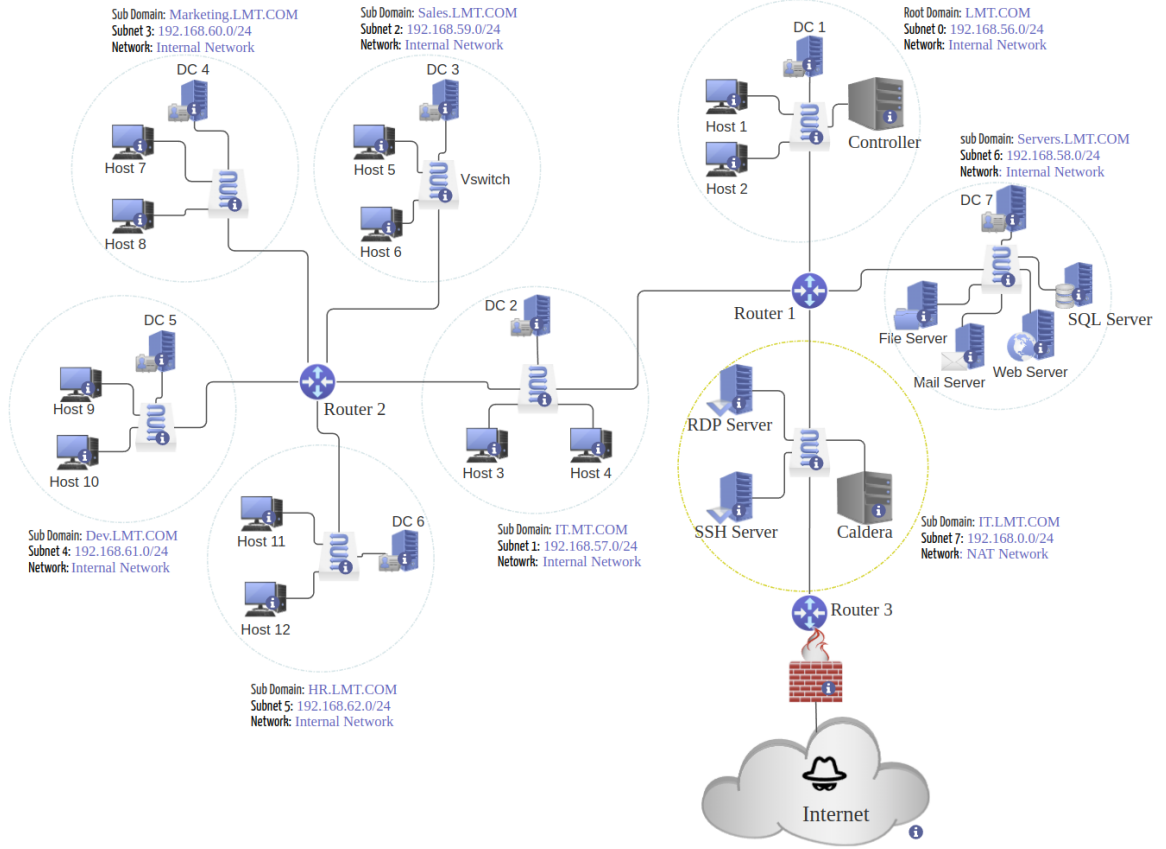


Fig. 3.3.1: The network topology used to generate LMDG dataset.

from external traffic, except the Demilitarized Zone (DMZ), which was configured as a NAT network. This NAT configuration allows the DMZ to communicate with external networks while maintaining the isolation of internal subnets, supporting a realistic simulation of enterprise network structures.

The experimental environment was configured with all Windows 10 and Windows 11 operating systems hosts, while servers operated on Windows Server 2022. This selection reflects commonly deployed systems in modern enterprise networks, ensuring the realism and relevance of the simulated environment for cybersecurity research.

This topology is realistic and superior to many commonly used topologies in the literature for several reasons. Firstly, it mirrors the complex, segmented network structure of a typical small to medium-sized enterprise, incorporating multiple subnets and dedicated Windows domains for different departments. This segmentation enhances security and reflects real-world organizational practices. Additionally, the

inclusion of a Demilitarized Zone (DMZ) for public-facing services and separate subnets for critical infrastructure such as company servers and root domains provide a more accurate and comprehensive environment for generating datasets. These elements contribute to a higher fidelity simulation of enterprise network traffic and potential security threats, making the datasets derived from this topology more applicable and valuable for the research community.

3.3.3 Benign Data Engine (BDE)

In the context of the LMDG framework, the Benign Data Engine (BDE) is tasked with generating normal network behavior, effectively simulating employee activities. Figure 3.3.2 provides an overview of the BDE engine, which comprises two primary components: the **Sessions Scheduler** 3.3.3.1 and the **Sessions Executor** 3.3.3.2. The engine operates based on four key inputs:

- **User Credentials and Hosts:** This includes the credentials of employees and the specific hosts (workstations or devices) they use within the network.
- **Sessions Scheduler Configuration File:** This file defines the parameters for the Sessions Scheduler, dictating how it should generate and manage sessions timing for each user or employee.
- **Behavioral Scripts:** These scripts detail the activities of employees, it can operate on both individual level and a departmental level, such as those specific to the IT department. They encapsulate routine tasks and behaviors expected in a typical workday.

The Sessions Scheduler orchestrates generating session behaviors (i.e., login and logout times), ensuring the simulated activities align with realistic standard user behavior patterns. Concurrently, the Sessions Executor enables the efficient simulation of multiple user sessions, reflecting the concurrent activities of various employees

within the network. This design enhances the generated data's realism and ensures scalability and performance in simulating complex network environments.

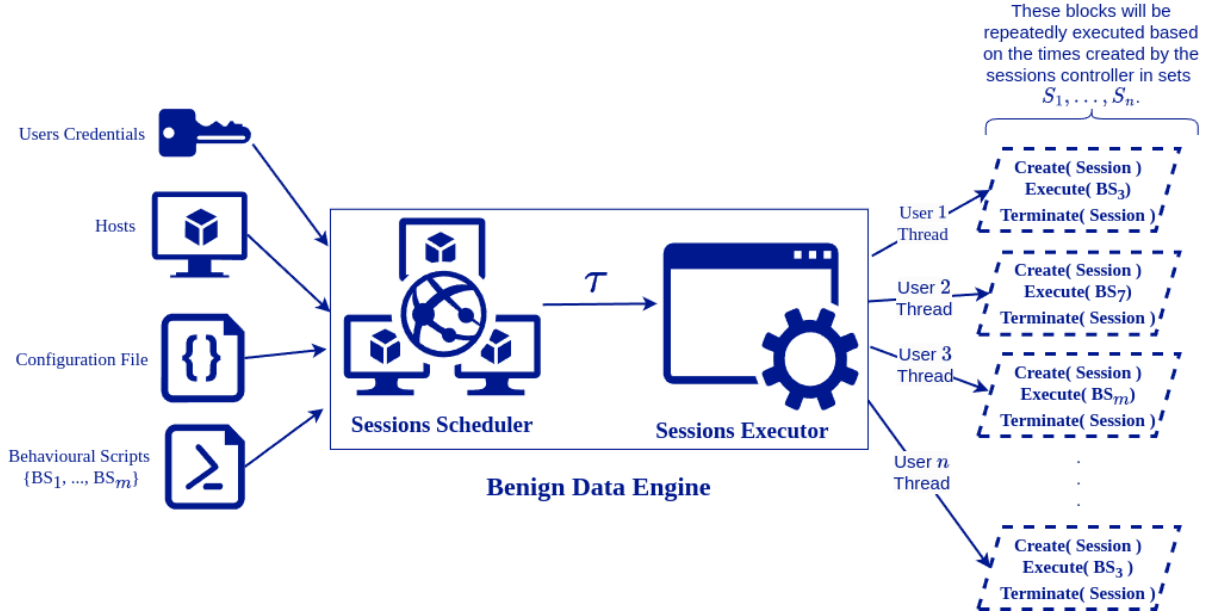


Fig. 3.3.2: Benign Data Engine (BDE) overview.

3.3.3.1 Sessions Scheduler

The role of the Sessions Scheduler is to generate a list of tuples

$$\mathcal{S}_i = [(t_1, t_2), (t_3, t_4), \dots, (t_{2k_i-1}, t_{2k_i})]$$

for each employee i , representing their session behavior. Each tuple (t_{2j-1}, t_{2j}) denotes the login and logout times of a session, where t_{2j-1} is the login time and t_{2j} is the logout time. The duration of a session is given by $t_{2j} - t_{2j-1}$. The final output of the Sessions Scheduler is a list of lists $\mathcal{T} = [\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n]$, where \mathcal{T} encapsulates the session behaviors for all n employees in the network. Each \mathcal{S}_i in \mathcal{T} provides a detailed account of an individual employee's login and logout activities throughout the day.

The process by which the Sessions Scheduler generates the lists \mathcal{S}_i for an employee i is outlined as follows. Initially, the Sessions Scheduler determines whether employee i is absent based on probability values specified in the configuration file (third input,

Figure 3.3.2) defined by the dataset creators. For instance, the dataset creators can define a probability interval $[p_1, p_2]$, where $0 \leq p_1, p_2 \leq 1$. The Sessions Scheduler then selects a random value from this interval to represent the probability of employee i being absent on a given day. This approach ensures that each employee i has a distinct probability of being absent. If employee i is absent, then the list $\mathcal{S}_i = \emptyset$. Additionally, there is a separate probability interval $[p'_1, p'_2]$ for determining absences during weekends, which typically corresponds to a higher probability of absence. The Sessions Scheduler selects a random value from this interval for weekends, reflecting the increased likelihood of employees being absent on non-working days.

If employee i is not absent, the Sessions Scheduler will proceed to generate the list \mathcal{S}_i . Initially, it determines the starting time (first login) for employee i . To facilitate this, the dataset creators define four time intervals representing various starting times: abnormally early, abnormally late, late, and on time. These intervals are denoted as $[t_{e1}, t_{e2}]$, $[t_{a1}, t_{a2}]$, $[t_{l1}, t_{l2}]$, and $[t_{o1}, t_{o2}]$ respectively. To determine the four possible starting times for the current employee i , the Sessions Scheduler randomly selects a value from each of the four corresponding intervals. Thus, for employee i , there exist four distinct candidate starting times denoted as $t_{\text{start_abnormal_early}}$, $t_{\text{start_abnormal_late}}$, $t_{\text{start_late}}$, and $t_{\text{start_on_time}}$. In the configuration file, operators can define different probability intervals for each possible starting time, namely $[p_{e1}, p_{e2}]$, $[p_{a1}, p_{a2}]$, $[p_{l1}, p_{l2}]$, and $[p_{o1}, p_{o2}]$. It is noteworthy that the probability intervals $[p_{e1}, p_{e2}]$ and $[p_{a1}, p_{a2}]$ are typically very small, reflecting the rarity of abnormally early and abnormally late starting times. Conversely, the interval $[p_{o1}, p_{o2}]$ is usually assigned the highest probabilities, indicating the likelihood of employees starting on time. Consequently, the Sessions Scheduler assigns a random probability value to each candidate starting time, drawn from their respective probability intervals. This process can be likened to tossing an unfair tetrahedron (a die with four faces), where each face represents a starting time option. The resulting face corresponds to the actual starting time of employee i , denoted as t_{start} , which constitutes the first value of the first tuple in the list \mathcal{S}_i , i.e., t_1 . Thus, the Sessions Scheduler effectively determines the starting time for employee i using this probabilistic method, ensuring that each potential starting

time is considered.

Selecting the end time t_{end} for each employee i , denoted as the second time in the last tuple of the list \mathcal{S}_i (i.e., last logout time t_{2k_i}), undergoes a process akin to determining the start time t_{start} . Similarly, the Sessions Scheduler employs a probabilistic approach, mirroring the methodology used for selecting t_{start} . Dataset creators define intervals representing various end times, such as abnormally early, abnormally late, late, and on time, each associated with corresponding probability intervals.

This probabilistic approach to determining employee starting and ending times offers flexibility and realism by modeling diverse punctuality behaviors. By assigning higher probabilities to on-time arrivals and accommodating randomness, it captures the dynamic nature of workplace scenarios. This method ensures the creation of diverse datasets, preventing predictable patterns and simulating unexpected occurrences. Ultimately, it enables the Sessions Scheduler to generate session behavior lists that closely resemble real-world employee activities, facilitating the creation of valuable network traffic datasets for applications such as security analysis, anomaly detection, and performance optimization.

The Sessions Scheduler is not limited to drawing values from the defined time and probability intervals using a uniform distribution; it can also utilize exponential and normal distributions. For instance, consider Figure 3.3.3, which illustrates the Sessions Scheduler's process of selecting the value for $t_{\text{start_abnormal_early}}$ over 20,000 iterations. In this example, the Sessions Scheduler is configured to draw a time value t within the interval [3:30 AM - 7:29 AM] according to an exponential distribution with a lambda $\lambda = 0.00037$, where λ is the distribution parameter. By plotting the frequency of each minute between 3:30 AM and 7:29 AM, Figure 3.3.3 demonstrates that the Sessions Scheduler successfully draws values in accordance with the specified exponential distribution. This capability allows for more realistic and varied simulations of employee behavior.

Figure 3.3.4 presents a similar experiment in which the Sessions Scheduler draws 20,000 values for $t_{\text{start_late}}$ based on a flipped exponential distribution. Conversely, Figure 3.3.5 depicts the Sessions Scheduler drawing 20,000 trials for $t_{\text{start_on_time}}$ according

to a normal distribution with a confidence interval of 2.58.

When conducting a large number of trials with the Sessions Scheduler, interesting patterns emerge in the selection of t_{start} times as an example. Each of the four potential start times— $t_{\text{start_abnormal_early}}$, $t_{\text{start_abnormal_late}}$, $t_{\text{start_late}}$, and $t_{\text{start_on_time}}$ —is drawn from distinct time intervals using different distributions. Specifically:

- $t_{\text{start_abnormal_early}}$ is drawn from the interval [3:30 AM - 7:29 AM] following an exponential distribution with $\lambda = 0.00028$.
- $t_{\text{start_abnormal_late}}$ is drawn from the interval [10:01 AM - 4:00 PM] following a flipped exponential distribution with $\lambda = 0.00020$.
- $t_{\text{start_late}}$ is drawn from the interval [8:31 AM - 10:00 AM] following a flipped exponential distribution with $\lambda = 0.00050$.
- $t_{\text{start_on_time}}$ is drawn from the interval [7:30 AM - 8:30 AM] following a normal distribution with a confidence interval of 2.58.

Each start time is then assigned a probability from predefined intervals:

- $t_{\text{start_abnormal_early}}$ is assigned a probability from the interval [0.025 - 0.05].
- $t_{\text{start_abnormal_late}}$ is assigned a probability from the interval [0.025 - 0.05].
- $t_{\text{start_late}}$ is assigned a probability from the interval [0.05 - 0.2].
- $t_{\text{start_on_time}}$ is assigned a probability of $1 - P(t_{\text{start_abnormal_early}}) - P(t_{\text{start_abnormal_late}}) - P(t_{\text{start_late}})$, which is at least 0.70.

The actual t_{start} is determined by a weighted random selection (unfair toss) among these four times. Running the Sessions Scheduler for 20,000 trials and plotting the histogram of t_{start} yields the distribution shown in Figure 3.3.6. As observed, the majority of occurrences fall within the 7:30 AM to 8:30 AM interval, following a normal distribution, due to $t_{\text{start_on_time}}$ having the highest probability. The next most frequent interval is 8:31 AM to 10:00 AM, following a flipped exponential distribution,

attributed to $t_{\text{start_late}}$ having the second highest probability. Finally, values at the extremes are rarely selected, reflecting their assigned low probabilities.

After defining t_{start} and t_{end} , the Sessions Scheduler will determine whether employee i will have a lunch break using a similar probabilistic approach. If a lunch break is scheduled, the controller will then specify $t_{\text{lunch_start}}$ and $t_{\text{lunch_end}}$, which denote the start and end times of the lunch break, respectively.

Algorithm 3.3.1 AllocatePoints

```

1: procedure ALLOCATEPOINTS( $t_s, t_e, P, i, m_{\min}, m_{\max}$ )
2:   if  $i = |P|$  then
3:     return  $(0, \emptyset)$ 
4:   end if
5:    $m \leftarrow \text{random}(m_{\min}, m_{\max})$ 
6:    $\Delta t \leftarrow t_e - 2m - t_s$ 
7:   if  $\Delta t < P[i]$  then
8:     return ALLOCATEPOINTS( $t_s, t_e, P, i + 1, m_{\min}, m_{\max}$ )
9:   end if
10:   $breaker \leftarrow \alpha$ 
11:  repeat
12:     $t_r \leftarrow \text{random}(t_s + m, t_e - m)$ 
13:     $\Delta t_1 \leftarrow t_e - P[i] - t_r$ 
14:     $\Delta t_2 \leftarrow t_r - t_s$ 
15:    if  $breaker = 0$  then
16:      return  $(0, \emptyset)$ 
17:    end if
18:     $breaker \leftarrow breaker - 1$ 
19:  until  $\Delta t_1 \geq m \wedge \Delta t_2 \geq m$ 
20:   $T \leftarrow (t_r, P[i])$ 
21:   $(S_1, T_1) \leftarrow \text{ALLOCATEPOINTS}(t_s, t_r, P, i + 1, m_{\min}, m_{\max})$ 
22:   $(S_2, T_2) \leftarrow \text{ALLOCATEPOINTS}(t_r + P[i], t_e, P, i + 1, m_{\min}, m_{\max})$ 
23:  if  $S_1 > S_2$  then
24:    return  $(1 + S_1, T_1 \cup \{T\})$ 
25:  else
26:    return  $(1 + S_2, T_2 \cup \{T\})$ 
27:  end if
28: end procedure

```

The final task of the Sessions Scheduler is to schedule the random logouts and logins occurring between t_{start} and $t_{\text{lunch_start}}$, as well as between $t_{\text{lunch_end}}$ and t_{end} .

This task is divided into two stages. First, the controller randomly determines the number of logouts before and after lunch and the duration of each logout. These values are chosen from intervals defined by the dataset creators in the configuration file (see Figure 3.3.2). Subsequently, the controller places these logouts on the timeline while adhering to specific rules, such as maintaining a minimum time interval between consecutive logouts and ensuring that logout times within the interval $[t_1, t_2]$ are proportional to the length of the interval. These constraints are also configurable parameters. The ‘AllocateLogouts’ algorithm 3.3.1 is devised to allocate logout points for employees within a designated time frame, with the aim of maintaining a minimum time interval between consecutive logouts. Beginning with the specification of a start time t_s and an end time t_e , the algorithm proceeds to recursively iterate through a list of predetermined periods P , each representing a duration for which a logout needs to be allocated. At each iteration, the algorithm dynamically determines a random minimum time between consecutive logouts, ensuring that this time does not violate the prescribed minimum and maximum thresholds m_{min} and m_{max} , α is a large number say 3000. By recursively branching into two segments—before and after each chosen logout point—the algorithm evaluates potential logout allocations while maximizing the number of allocated points. Through this recursive process, the ‘AllocatePoints’ algorithm optimally balances randomness with adherence to time constraints, ensuring efficient and effective logout point allocation.

3.3.3.2 Sessions Executor

The second component of the Benign Data Engine (Figure 3.3.2) is the Sessions Executor. This component is responsible for creating a thread or job for each employee i in the list \mathcal{T} . Utilizing the employee’s credentials, the Sessions Executor executes the corresponding behavioral script BS_j on the specified host H_r at the scheduled session times listed in S_i .

For the thread associated with employee i , at the start time t_{2j-1} from the tuple (t_{2j-1}, t_{2j}) in the list S_i , the Sessions Executor initiates a remote session on host H_r using the credentials of employee i . It then begins executing the behavioral script

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

BS_j linked to that employee. The script continues to run until the end time t_{2j} , at which point the session is terminated. This process repeats for each subsequent tuple (t_{2j+1}, t_{2j+2}) in S_i until the final tuple (t_{2k_i-1}, t_{2k_i}) is reached.

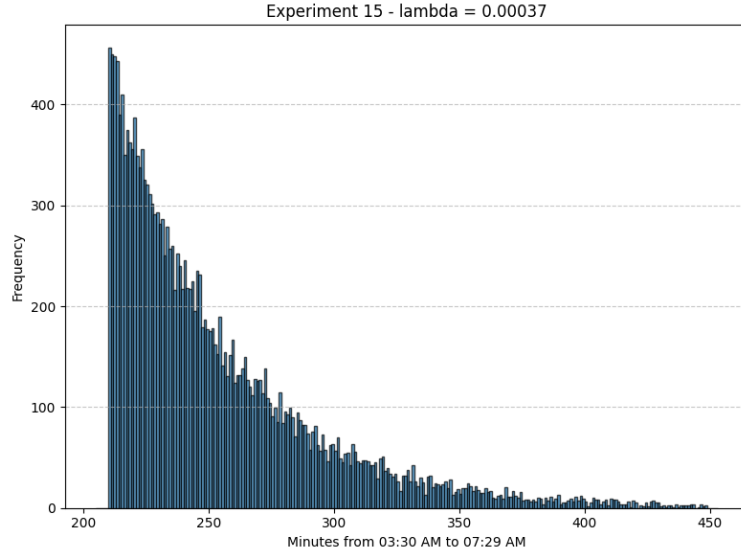


Fig. 3.3.3: Frequency distribution of $t_{\text{start_abnormal_early}}$ from 03:30 AM to 07:29 AM over 20,000 trials. The distribution follows an exponential distribution with a rate parameter $\lambda = 0.00037$, indicating higher frequencies of abnormal early start times occurring at earlier minutes and tapering off towards later minutes.

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

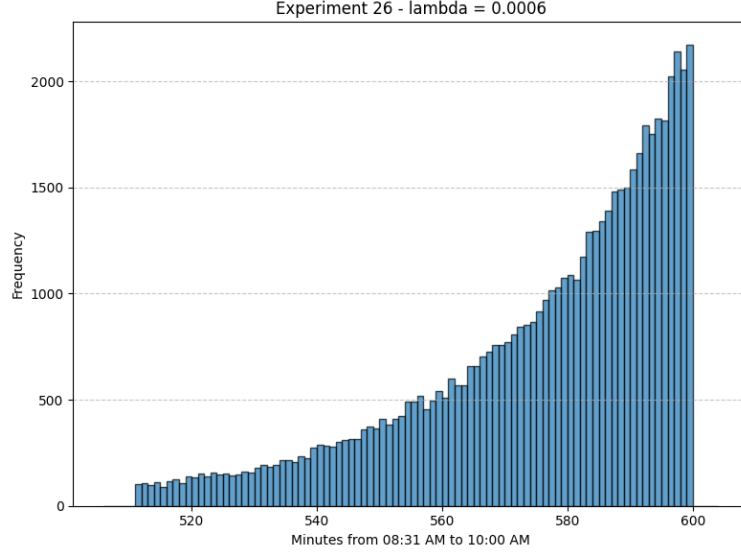


Fig. 3.3.4: Frequency distribution of $t_{\text{start_late}}$ from 08:31 AM to 10:00 AM over 20,000 trials. The distribution follows a flipped exponential distribution with a rate parameter $\lambda = 0.0006$, demonstrating lower frequencies at earlier times and gradually increasing towards later times.

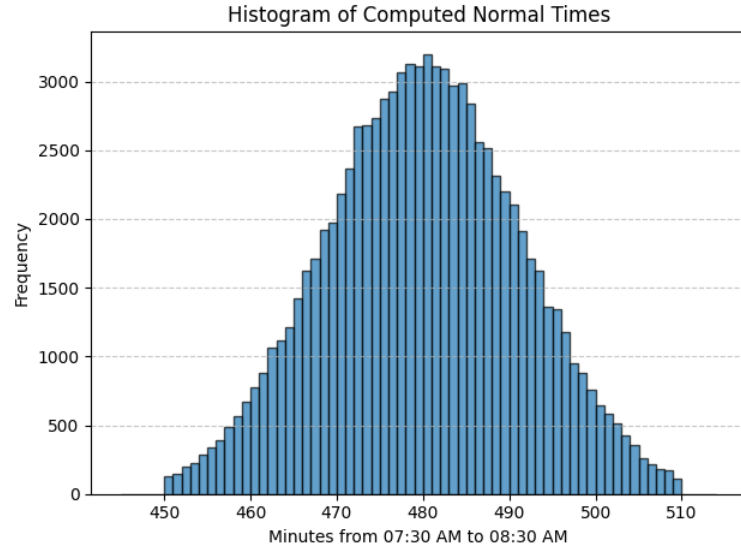


Fig. 3.3.5: Frequency distribution of every minute from 07:30 AM to 08:30 AM of $t_{\text{start_on_time}}$ over 20,000 trials drawn by the Sessions Scheduler based on a normal distribution with confidence interval of 2.58.

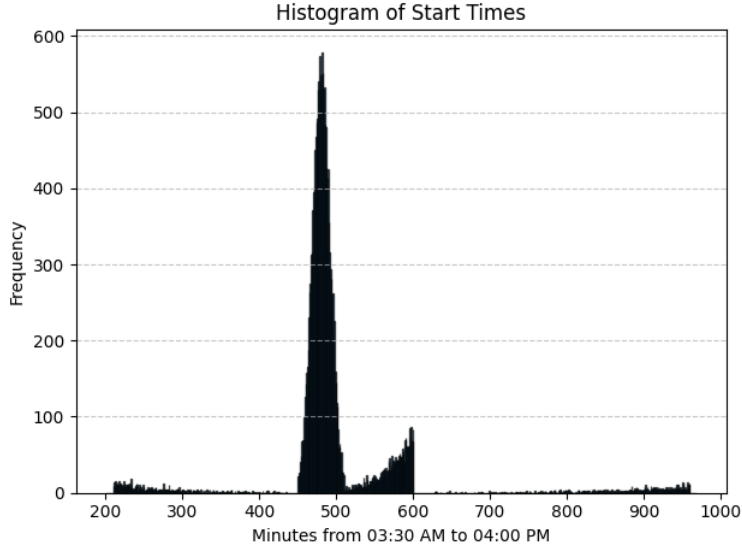


Fig. 3.3.6: Frequency distribution of every minute from 03:30 AM to 04:00 PM of t_{start} over 20,000 trials drawn by the Sessions Scheduler.

Each department can have a single behavioral script that represents the general behavior of its employees. Alternatively, there is an option for each user to have their own individual behavioral script, configurable in the configuration file. For every execution block (Figure 3.3.2), a subset of the available behaviors in the behavioral script will be executed at random using a probabilistic approach. These behaviors simulate typical user actions during a normal working day, such as browsing, downloading from the internet, running local programs, accessing internal servers (file server, database server, web server). The behavioral scripts within the framework can be adapted to align with various enterprise-specific use cases, allowing for a high degree of customization to mirror distinct operational environments. These scripts can also be configured to execute additional programs and activities that more accurately emulate realistic user behaviors, enhancing the fidelity of the simulation.

The described approach offers a realistic and flexible method for simulating user behavior in a network environment. By using customizable behavioral scripts for departments or individual users, and employing a probabilistic approach to execute a variety of typical user actions, the system ensures dynamic and varied simulations. The approach's scalability and adaptability make it suitable for simulating both small

and large sized organizations. The source code for BDE has been made available on our GitHub [56], providing researchers with access for purposes of dataset generation or other academic and research pursuits.

3.3.4 Attack Engine (AE)

In this context, an "attack engine" refers to a method or framework that enables the automated execution of cyberattacks. For example, automating DDoS attacks can often be achieved by deploying specific scripts on the attacking hosts to initiate the attack. However, as discussed in this section, automating lateral movement attacks presents unique challenges that are more complex and less straightforward than those associated with simpler scripted attacks.

3.3.4.1 Lateral Movement Attacks

According to the MITRE ATT&CK framework [59], nine tactics qualify as lateral movement techniques. These include *Exploitation of Remote Services*, *Internal Spearphishing*, *Lateral Tool Transfer*, *Remote Service Session Hijacking*, *Remote Services*, *Replication through Removable Media*, *Software Deployment Tools*, *Taint Shared Content*, and the *Use of Alternate Authentication Material*. In the LMDG dataset, multiple lateral movement tactics from this list—such as *Exploitation of Remote Services* and the *Use of Alternate Authentication Material*—are employed, as discussed further below 3.3.4.5.

Each of these lateral movement tactics encompasses various techniques. For instance, the "Use of Alternate Authentication Material" tactic can be executed through techniques like "Pass-the-Hash" or "Pass-the-Ticket" attacks. To clarify the complexity and unique nature of lateral movement attacks compared to more straightforward attack types, we provide a detailed example of one of these attacks. This analysis highlights the operational challenges and automation complexities inherent in implementing these advanced tactics.

One of the attack scenarios demonstrated in the LMDG dataset involves a *pass-the-hash (PtH)* attack, a technique classified under the “Use of Alternate Authentication Material” tactic in the MITRE ATT&CK framework. An outline of the attack sequence is depicted in Figure 3.3.7. The scenario begins by assuming an attacker has obtained the local administrator credentials for domain controller DC2 in subnet 1, potentially through techniques like phishing. Using these credentials, the attacker initiates an SSH connection to an SSH server in subnet 7 (attack step 1 in Figure 3.3.7) and subsequently connects to DC2 (attack step 2 in Figure 3.3.7) via SSH using the same credentials. Once on DC2, the attacker downloads and executes *Mimikatz* to extract credential hashes from the LSASS process, including those from recent sessions. In this case, an enterprise administrator recently accessed DC2 (shown by the green arrow in Figure 3.3.7), allowing the attacker to retrieve the administrator’s credentials. The attacker gains an elevated shell with the enterprise admin hash (step 3 in Figure 3.3.7), enabling access to restricted directories on a file server in subnet 6 (step 4 in Figure 3.3.7). This elevated access allows sensitive information to be exfiltrated from a folder accessible only to the enterprise administrator. Attack step 3 in figure 3.3.7 represents a transition from one privilege level to another (privilege escalation), which aligns with our proposed definition of lateral movement as outlined in the Discussion section 3.7. We classify this privilege escalation as a form of pivoting, categorizing it within lateral movement activities.

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

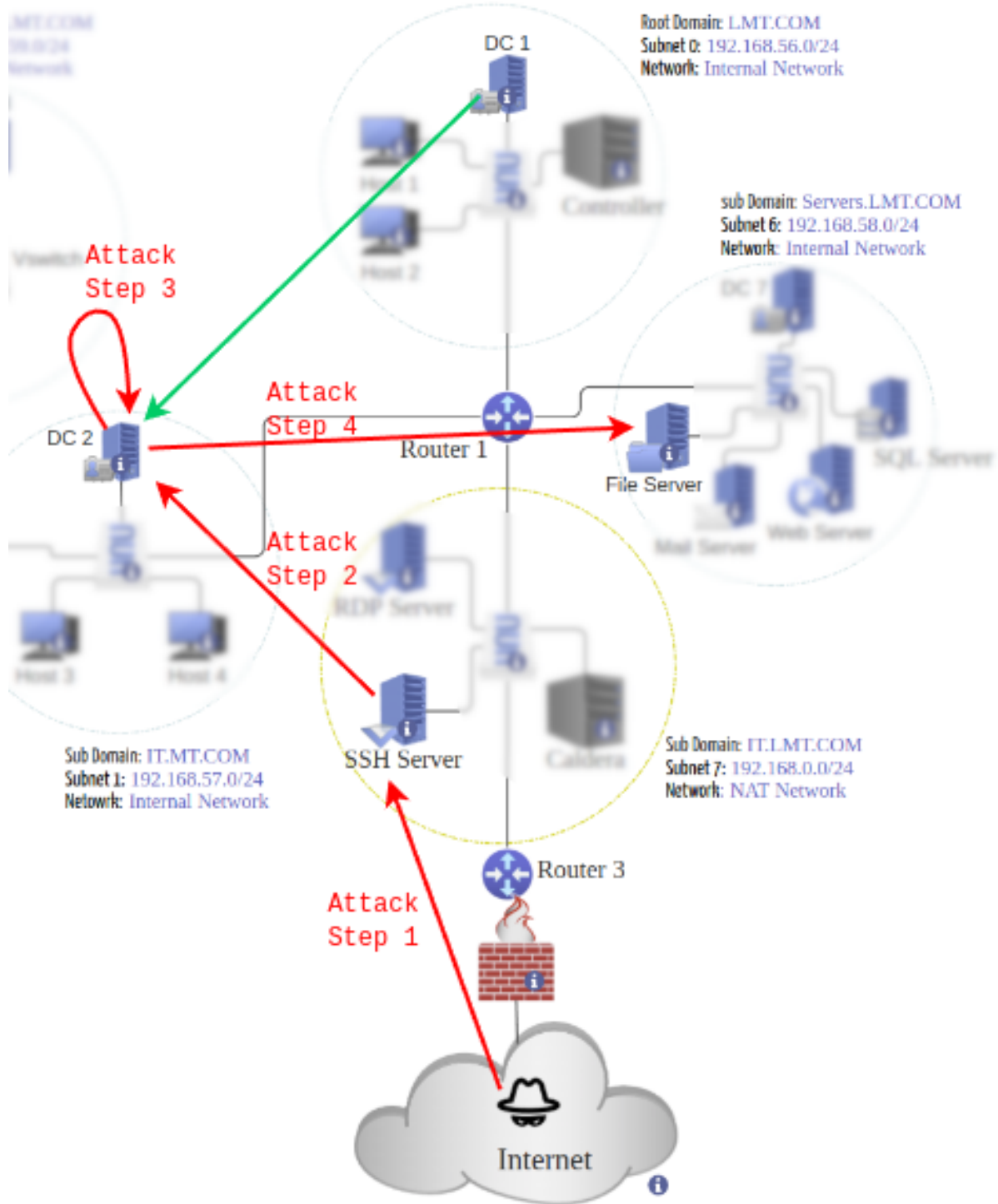


Fig. 3.3.7: First Attack Scenario in LMDG dataset which is a Passing the Hash attack (PtH). This figure also illustrates the traversal behavior (hops) in the Passing the TGT attack scenario outlined in Subsection 3.3.4.5.1, providing a step-by-step visualization of the movement through network nodes.

3.3.4.2 Challenges in Automating Lateral Movement Attacks

As demonstrated in the preceding attack example 3.3.4.1 and further detailed in subsection 3.3.4.5, obtaining elevated and reverse shells frequently occurs in executing lateral movement attacks. This section addresses the unique challenges this poses for automating lateral movement attacks.

Consider automating the *Pass-the-Hash* (PtH) attack described in subsection 3.3.4.1 and illustrated in Figure 3.3.7. Initial steps, such as SSH-ing into the SSH server with stolen credentials, then SSH-ing again to the domain controller DC2 to execute commands like downloading and running Mimikatz, are feasible to automate using conventional scripting methods. These represent Attack Steps 1 and 2 in Figure 3.3.7.

However, challenges emerge when performing the PtH attack to obtain an elevated shell via Mimikatz. The command for PtH, shown in Figure 3.3.8, involves “sekurlsa::pth,” specifying Mimikatz to perform PtH, followed by required parameters like user, domain, NTLM hash, and executable command (in this case, to spawn an elevated shell). Even if the NTLM hash is known, running this command within an automation script initiates a new *cmd.exe* process under elevated credentials, making it difficult, if not impossible, to access or interact with this newly created shell. Identifying and interfacing with this elevated shell becomes highly challenging without access to properties like process ID, particularly if the current credentials lack the necessary permissions.

This issue is compounded when dealing with reverse shells, as the new shell may be spawned on a different host. Furthermore, automation becomes more complex if the NTLM hash is unknown and must be dynamically retrieved during attack execution. Automating lateral movement thus requires adapting to dynamically derived information, such as reusing information from previous attack steps—something traditional automation scripts are not well-equipped to handle.

Two significant challenges arise in automating lateral movement attacks through traditional methods: handling elevated and reverse shells and dynamically extracting

and reusing information from prior attack steps for subsequent actions.

```
1 sekurlsa:pth /user:Administrator /domain:LMT
   ↪ /ntlm:cb8a428385459087a76793010d60f5dc /run:"cmd.exe"
```

Fig. 3.3.8: Example command for obtaining an elevated shell using the pass-the-hash (PtH) technique via *Mimikatz*.

3.3.4.3 A Candidate Solution

A viable solution to the challenge of automating elevated and reverse shells in lateral movement (LM) attacks, as discussed in section 3.3.4.2, is to implement a client-server architecture to coordinate different attack steps. This approach utilizes an attack controller, or orchestrator, that issues commands to agents deployed on various hosts. These attack agents act as terminals, receiving instructions from the orchestrator. To illustrate, we will revisit the attack scenario in section 3.3.4.1 and demonstrate how this client-server architecture could be applied to streamline the automation of the attack.

Consider an orchestrator or controller deployed in subnet 7, with an attack agent, A_1 , running on the SSH server—equivalent to the first attack step illustrated in Figure 3.3.7, analogous to an external SSH connection to the SSH server. Instead of establishing an SSH connection from agent A_1 to DC2, a new agent, A_2 , can be spawned on DC2 using the same credentials, representing the second attack step in Figure 3.3.7. On DC2, agent A_2 can execute the same commands detailed in subsection 3.3.4.1, such as downloading and running *Mimikatz*.

The third step in the attack sequence involves a pass-the-hash (PtH) operation to obtain an elevated shell, or terminal, that will access the file server. Instead of executing the PtH command (Figure 3.3.8) on A_2 to spawn a new, isolated terminal, we can modify the PtH command's `/run` parameter to spawn an elevated agent A_3 . Instead of `/run:"cmd.exe"`, the parameter would be configured as `/run:"<code to spawn agent A_3 >"`. After executing the modified PtH command, the elevated agent A_3 is created, completing the final attack step (step 4 in Figure 3.3.7) to access a

restricted directory and exfiltrate data.

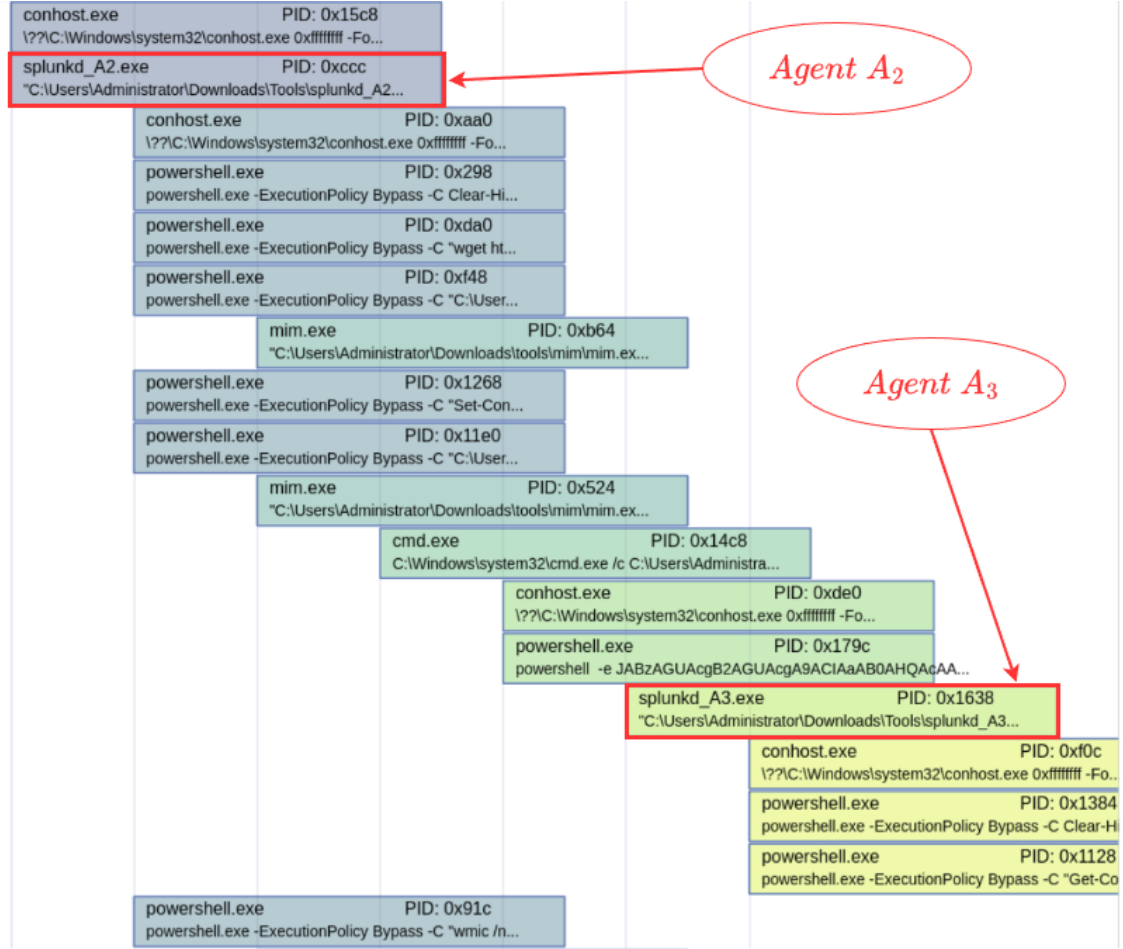


Fig. 3.3.9: Partial process tree illustrating the execution of the Pass-the-Hash (PtH) attack on DC2, as discussed in subsection 3.3.4.1.

Figure 3.3.9 illustrates a portion of the process tree generated during the execution of the Pass-the-Hash (PtH) attack using a client-server architecture for automation. In this configuration, we observe Agent A_2 , represented by the process `splunkd_A2.exe`, which is the initial agent deployed on Domain Controller DC2. Agent A_2 is responsible for executing several preliminary commands. Subsequently, it performs the PtH attack, leveraging elevated privileges to spawn a new, elevated agent, Agent A_3 , identifiable by the process name `splunkd_A3.exe`. After its creation, Agent A_3 will start communicating with the controller to continue to execute the attack sequence, ultimately accessing a restricted directory on the file server to retrieve sensitive data. This process tree visualizes the hierarchical structure of process inter-

actions, illustrating how the client-server model facilitates controlled privilege escalation within the attack flow. This approach effectively addresses the challenge of managing elevated and reverse shells. Any newly spawned agent—whether on the same host as its parent agent or on a different host in the case of reverse shells—establishes communication with the central controller, enabling it to receive and execute required commands. We refer to this valuable feature as ***attack steps chaining***, a capability that can be achieved with any client-server-based attack orchestration tool, several tools with this functionality, including *CALDERA*, are mentioned in [33].

3.3.4.4 CALDERA as an Attack Engine

Following the approach outlined in [33], we utilized Caldera to implement the client-server architecture discussed in subsection 3.3.4.3 to automate attack execution steps. As previously noted, Caldera, along with other tools referenced in [33], can be employed to facilitate this level of attack automation. We refer to such tools collectively as "attack engines."

Caldera™ [60] is an adversary emulation platform developed by MITRE for autonomous breach-and-attack simulations, manual red-team operations, and automated incident response. Based on the MITRE ATT&CK™ [59] framework, Caldera includes a core system consisting of the main framework code, an asynchronous command-and-control (C2) server, a REST API, and a web interface. It also supports plugins—separate repositories that extend the core functionality by adding agents, graphical interfaces, and collections of Tactics, Techniques, and Procedures (TTPs), enabling a flexible and comprehensive approach to adversary emulation.

We now examine the various components of Caldera and demonstrate how it can be leveraged to automate attack execution for the generation of cybersecurity datasets, using examples from the attack scenario described in subsection 3.3.4.1.

The primary component of Caldera is the ***Caldera Server***, an asynchronous command-and-control (C2) server equipped with a REST API and a web interface. Upon deployment, each agent communicates with the Caldera Server to report its status (e.g., alive or inactive) and to receive and execute commands. Most automation

tasks are managed through the Caldera Server’s web interface. It is essential to clarify that, in this context, the Caldera Server is employed exclusively as a tool for automating attack execution specifically for cybersecurity dataset creation rather than as an actual C2 server. Ideally, the Caldera Server should operate without leaving any trace of its communication with the agents or other artifacts that could influence the dataset. The server should remain transparent and invisible to avoid contaminating the dataset with automation-related traces. Attack scenarios that involve an actual C2 server should be hosted on a separate machine from the Caldera Server, and all Caldera Server artifacts should be removed from the dataset to maintain its integrity.

The second key component of the Caldera framework is the **Agents** deployed on various hosts to perform specific attack steps. These Caldera agents function as terminals or shells, receiving preconfigured commands—known as “abilities” within the Caldera framework—and executing them on the target host. The Caldera server can generate agent code that establishes initial access when executed on a target host with specific credentials and marks the beginning of an attack sequence. This agent code contains essential information, including the Caldera server’s IP address and the group to which the agent belongs, which plays a critical role in attack steps chaining (as will be detailed in the discussion of Caldera Operations below). In the context of the Caldera framework, it is assumed that an initial compromise has already been achieved. For instance, in the attack detailed in Subsection 3.3.4.1 and illustrated in Figure 3.3.7, we posit that the initial attack step—specifically, establishing an SSH connection to the internal network—has been successfully executed. The first agent, Agent A_1 , is deployed on the SSH server in this scenario. Subsequently, Agent A_1 autonomously executes the remaining attack steps as directed by the controller. This process culminates in the spawning of a new agent, designated as Agent A_2 , on DC2. Agent A_2 will further advance the attack sequence and facilitate the creation of an elevated agent, Agent A_3 . Both agents are depicted in Figure 3.3.9. This sequence of actions is what we refer to as **attack steps chaining** in Subsection 3.3.4.3.

Next, we discuss the concepts of **Abilities** and **Adversaries** within the Caldera framework. An **Ability** represents one or more commands designed to execute a

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

specific action and serves as an atomic building block for attacks. For example, an ability may execute a simple command, such as a 10-second sleep, or a more complex command, such as Mimikatz execution. Each ability is defined by a name, description, and associated MITRE ATT&CK tactic and technique, which will help in the labeling process as discussed in section 3.3.5. An **Adversary**, on the other hand, comprises a sequence of abilities that together model a specific behavior, often emulating the attack patterns of well-known APT groups. Figure 3.3.10 illustrates an example of an adversary along with its associated abilities, where each row represents a specific ability. We can now construct our attack steps as adversaries, as depicted in Figure 3.3.10, where this particular adversary models attack step 3 from the scenarios outlined in Section 3.3.4.1 and visualized in Figure 3.3.7. The Caldera framework offers substantial flexibility in designing and executing custom attack scenarios, enabling precise representation of complex adversarial behaviors.

Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup
1	LMT_SC1-3_Download Mimikatz	execution	Command and Scripting Interpreter: PowerShell	Windows				×
2	LMT_SC1-4_Mimikatz	execution	Command and Scripting Interpreter: PowerShell	Windows		🔑		×
3	LMT_SC1-5 Create agent.bat file	execution	Command and Scripting Interpreter: PowerShell	Windows				×
4	LMT_SC1-6_Pass the Hash	<u>lateral-movement</u>	Use Alternate Authentication Material: Pass the Hash	Windows				×

Fig. 3.3.10: Caldera Adversary representing attack step 3 from figure 3.3.7 . Each row corresponds to an ability, detailing the command name and associated MITRE ATT&CK tactics and techniques.

Next, we discuss Caldera **Operations**, which involves assigning specific adversary profiles to active agents and prompting them to execute these profiles. In our context, these profiles correspond to distinct attack steps, such as the adversary shown in Figure 3.3.10, representing the third attack step illustrated in Figure 3.3.7. Each agent is assigned a group designation, and execution begins for adversaries within selected groups; thus, any agent within the specified group will initiate the assigned adversary profile, even if only a single agent is present.

Finally, we examine Caldera **Parsers** and **Facts**, essential features that address the second challenge outlined in Subsection 3.3.4.2: dynamically extracting and

reusing information from prior attack steps for subsequent actions. When an agent executes a command—such as dumping NTLM hashes—parsers extract relevant information from the command output, such as usernames and associated NTLM hashes. This data is then transmitted back to the Caldera server, where it is stored as **Facts** for subsequent attack steps, enabling actions like pass-the-hash attacks with stored user credentials. Thus, Facts are live variables that carry information across different attack stages. Facts are generated in two primary ways: dynamically from parsers or manually, such as by saving a specific user’s NTLM hash to be used later in the attack sequence.

3.3.4.5 Lateral Movement Attacks in LMDG Dataset

The LMDG dataset contains seven attack scenarios that achieve lateral movement using various tactics and techniques. Of these, three attacks were successful, while four were unsuccessful. We discuss possible reasons for each unsuccessful attack, considering that our setup includes Windows 10, Windows 11, and Windows Server 2022—the latest Windows versions with advanced security mechanisms. This combination of successful and unsuccessful attacks is valuable for understanding attacker behavior, as many attacks tend to fail due to robust defenses, with only some achieving success.

Our dataset includes multiple versions of each attack, targeting different hosts and subnets. In some cases, attacks were executed repeatedly to enrich the dataset with diverse instances of attack records; we refer to this repetition of the same scenario, version pair, as a trial.

The attack steps depicted in the figures, e.g., figure 3.3.7, within the attack explanations represent lateral movement hops, as defined in Section 3.7. All attack scenarios share the first two steps: initial access to the SSH server from outside the network using stolen credentials, followed by access to an additional internal machine. Beyond these initial steps, each attack scenario diverges in tactics and execution. More details about attacks execution are presented in 3.4.

3.3.4.5.1 Second Attack Scenario The first attack scenario was introduced in Subsection 3.3.4.1. In the second scenario of the LMDG dataset, a Pass-the-TGT (Ticket Granting Ticket) attack is presented, corresponding to the "Use of Alternate Authentication Material" tactic in the MITRE ATT&CK framework. The attack unfolds through several stages, as depicted in Figure 3.3.7.

The process begins with the attacker SSH-ing into the SSH server located in subnet 7 using stolen credentials of the local administrator from the IT department in subnet 1 (Step 1). The attacker then proceeds to SSH into the domain controller (DC2) in subnet 1 using the same credentials (Step 2). The enterprise administrator, shown in Figure 3.3.7 with a green arrow, has recently authenticated to DC2. The attacker then dumps the LSASS memory on DC2, exposing authentication tokens, including the enterprise administrator's TGT. In Step 3, the attacker injects this TGT into memory, impersonating the enterprise administrator and obtaining elevated privileges without requiring the administrator's plaintext credentials. Finally, in Step 4, the attacker uses the injected TGT to access the restricted file server in subnet 6, extracting sensitive information from directories that are typically only accessible to the enterprise administrator.

The attack ultimately failed, and it may have failed due to several factors. Kerberos replay protection mechanisms could have blocked the reuse of the TGT, while session constraints tied to specific network contexts may have prevented the attacker from using the ticket across subdomains. Additionally, domain policies and advanced auditing configurations might have detected and blocked unusual ticket requests. Although the attack successfully obtained the TGT in Step 3, these barriers may have prevented further progress to Step 4, where the attacker would have attempted to access the restricted file server and exfiltrate data. This scenario highlights potential challenges in lateral movement through authentication token manipulation.

3.3.4.5.2 Third Attack Scenario In another scenario within the LMDG dataset, an AS-REP Roasting attack demonstrates a technique aligned with the Credential Access tactic in the MITRE ATT&CK framework. As detailed in Figure 3.3.11, the

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

attack sequence begins with an attacker who initially gains access to the SSH server using the credentials of an employee in the marketing department in subnet 3 (step 1). then the attacker moves to host seven within subnet three via SSH (step 2). With access established, the attacker downloads necessary tools, including Rubeus, to facilitate the AS-REP Roasting attack.

AS-REP Roasting (also known as "ASREPRoast") exploits a vulnerability in Active Directory (AD) accounts configured without requiring Kerberos pre-authentication. This lack of pre-authentication allows an attacker to request an encrypted Ticket Granting Service (TGS) response directly, exposing the password hash, which can then be extracted and cracked offline to reveal plaintext credentials.

In this scenario, the attacker identifies an AS-REP Roasting user within the domain, extracts the hash, and successfully cracks it offline. With the recovered credentials, the attacker moves laterally within the network using the newly obtained user account (step 3). After establishing access under this new identity, the attacker requests a Ticket Granting Ticket (TGT) with delegation rights. This TGT is saved locally, and the attacker uses PowerShell to extract the Base64-encoded portion of the ticket, storing it as `ticket.base64`. Leveraging Rubeus again, the attacker initiates Service-for-User (S4U) impersonation to escalate privileges by impersonating the Administrator user for a specific service (in this case, HTTP/service within subnet 3). This impersonation uses the S4U2Self and S4U2Proxy Kerberos extensions, allowing the attacker to exploit Kerberos delegation mechanisms. By injecting this ticket into memory, the attacker aims to achieve Pass-the-Ticket (PTT) authentication, gaining unauthorized access to target resources with elevated privileges (step 4), an advanced privilege escalation technique in Kerberos-enabled environments.

The attack has failed, and it may have failed due to several technical constraints, including strict KDC and domain policies enforced by the Windows Server environment, which restrict TGT delegation and service account impersonation. For example, if the Service-for-User-to-Proxy (S4U2Proxy) functionality were disabled, the impersonation attempt would likely fail. Additionally, advanced domain monitoring and auditing could detect and block abnormal Kerberos requests in real time. The

network's use of Kerberos Constrained Delegation (KCD) may have also limited the attacker's ability to delegate to the HTTP service in subnet 3, preventing the impersonation attempt. These protective measures might have halted the attack at step 4, preventing further privilege escalation.

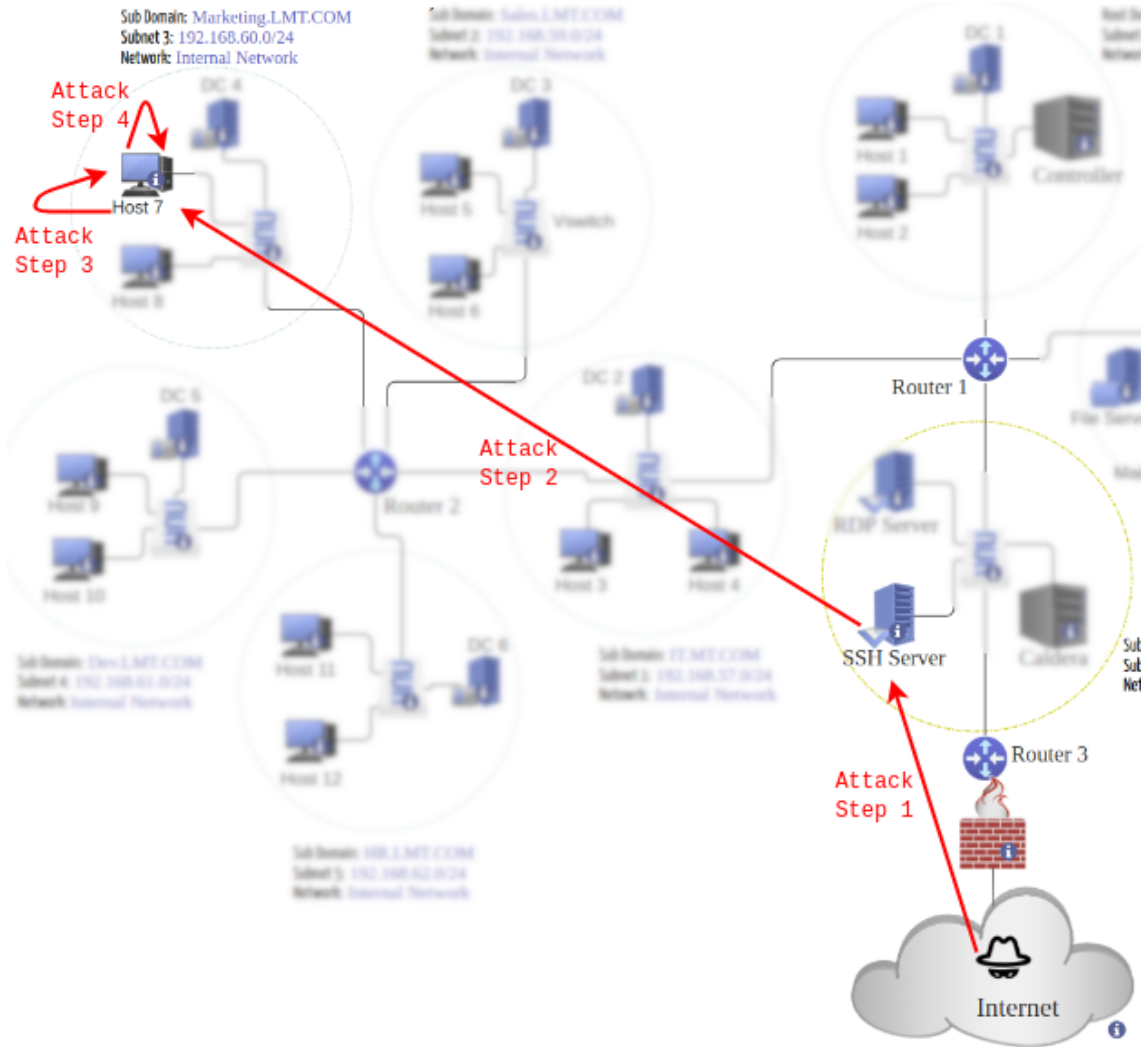


Fig. 3.3.11: Third Attack Scenario in LMDG dataset which is AS-REP Roasting attack.

3.3.4.5.3 Fourth Attack Scenario In another scenario from the LMDG dataset, an advanced delegation attack demonstrates multiple techniques spanning Credential Access, Persistence, and Privilege Escalation tactics within the MITRE ATT&CK framework. The sequence, detailed in Figure 3.3.12, begins with an attacker who initially gains access to an SSH server using the credentials of a standard user (step 1).

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

The attacker then transitions laterally to host nine within subnet four via SSH (step 2). With access to this host, the attacker downloads critical tools, such as Rubeus, necessary to facilitate an AS-REP Roasting attack.

In this scenario, the attacker identifies a domain user vulnerable to AS-REP Roasting, extracts the hash, and cracks it offline. With the recovered credentials, the attacker conducts lateral movement by using this new user account to further their access within the network (step 3).

Upon acquiring additional access, the attacker identifies a misconfiguration involving the AddSelf permission under the current user account. This permission enables the attacker to assign themselves to AD groups with higher privileges. Exploiting this, the attacker leverages the AddSelf permission to add their account to the Exchange Windows Permissions group, which has broad rights within AD (step 4). The attacker now has a foothold to manipulate permissions further and moves to escalate privileges.

With these elevated permissions, the attacker uses PowerView (from the PowerSploit toolkit) to modify AD object permissions. By executing the Add-DomainObjectAcl command, the attacker assigns themselves DCSync rights over the Domain Admins group by altering the Access Control List (ACL) (step 5). The DCSync right is a highly privileged permission that allows the account to perform directory replication activities, effectively emulating domain controller operations. This privilege enables the attacker to request password hashes and other sensitive data directly from the domain controller, granting access to high-privilege accounts, including domain administrators. This escalation step gives the attacker control of the domain without direct access to the domain controller.

Following this setup, the attacker performs a credential theft attack using Mimikatz, invoking the lsadump::dcsync command to impersonate a domain controller and retrieve sensitive authentication data, precisely the NTLM hash of the domain administrator within subnet 4. This process leverages the granted DCSync rights to replicate AD data without compromising the domain controller. By capturing the NTLM hash, the attacker can impersonate the administrator, potentially carrying

out Pass-the-Hash (PtH) attacks to gain unauthorised access across the domain.

In step 6, the attacker conducts the PtH attack to gain domain administrator access. However, due to the security restrictions related to double-hop authentication, the attack fails to propagate the administrator's full rights across machines. To overcome this, the attacker reinitiates the attack in step 6 within the same machine context, renewing the Ticket Granting Ticket (TGT) to ensure persistence and full administrator rights.

Finally, in step 7, with administrator privileges secured, the attacker can access sensitive data on restricted file servers within subnet 6, demonstrating a complete compromise and data exfiltration pathway within the network. This scenario highlights the critical risk posed by unauthorised delegation and improperly configured permissions, especially when AddSelf and DCSync rights are combined to achieve unauthorised privilege escalation and persistence within an AD environment.

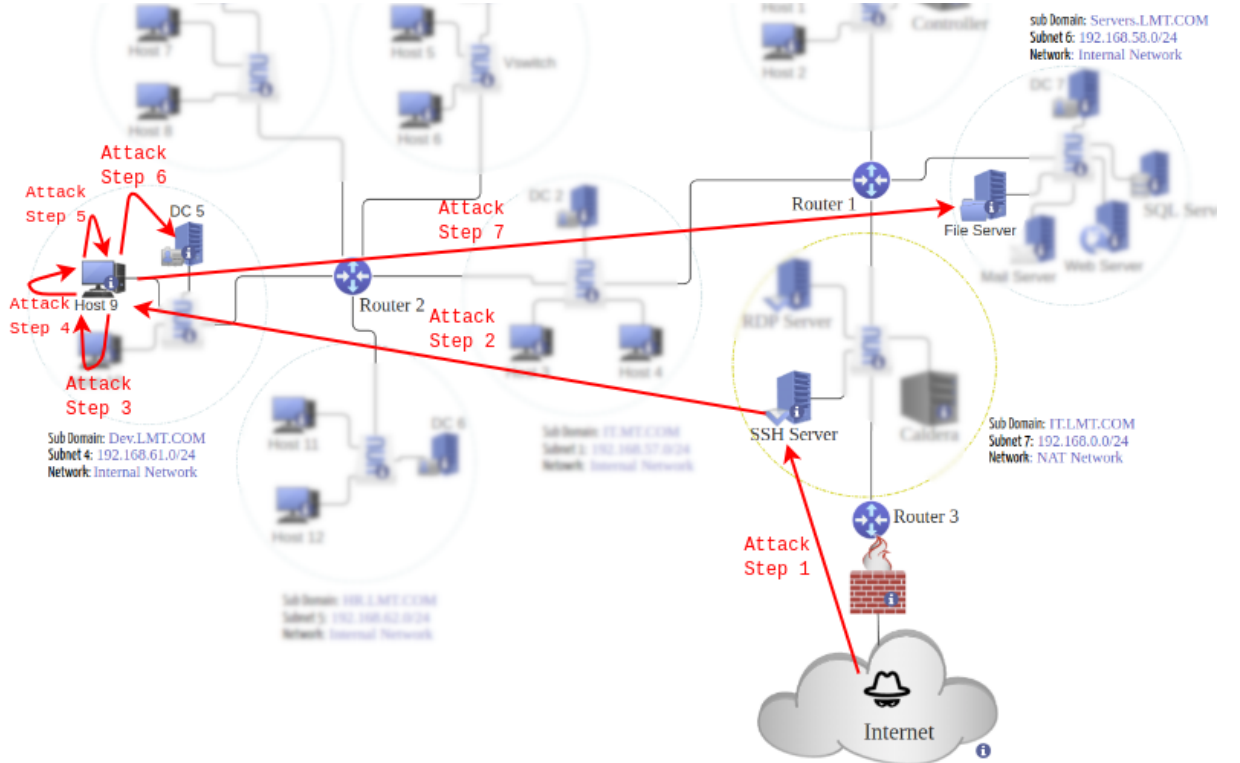


Fig. 3.3.12: Fourth Attack Scenario in LMDG dataset which is an Advanced Delegation attack.

3.3.4.5.4 *fifth Attack Scenario* In another scenario from the LMDG dataset, a password spray attack demonstrates multiple techniques spanning Credential Access, Persistence, Privilege Escalation, and Exfiltration tactics within the MITRE ATT&CK framework. The sequence, detailed in Figure 3.3.13, begins with an attacker who initially gains access to an SSH server using the credentials of a standard user (step 1). The attacker then transitions laterally to host seven within subnet three via SSH using the same credentials (step 2). With access to this host, the attacker finds a zip file protected with a password. The attacker will then download a wordlist that he will need to perform his brute attack, forcing the zip file using an automated custom script, where he succeeds. The attacker identifies the users in the domain and then performs a password-spraying attack where he finds a valid combination of credentials to be the domain administrator on subnet 3.

In step 3, the attacker will log in with the found credentials, perform privilege escalation, gaining all the rights on the subnet three domain with an administrator account on the DC 4 machine. The attacker looks for writable shares where he finds one owned by the enterprise administrator where a script is present and is being executed by the enterprise administrator periodically. The attacker abuses the privilege of the owned account to write into this script, adding a malicious payload to take ownership of the enterprise admin with a reverse shell payload. A reverse shell in cybersecurity is a network connection in which a compromised system initiates an outbound connection to a remote attacker-controlled machine, creating a shell access session. Unlike a standard shell, where an attacker attempts to connect directly to the target system, a reverse shell "reverses" the connection flow. The compromised system opens a specific port when the reverse shell is executed in step 4. It connects to the attacker's machine, typically via protocols like TCP or HTTP, gaining access over the enterprise administrator account, compromising the whole domain. Finally, in step 5, the attacker can now access the forbidden data in the file server in subnet 6.

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

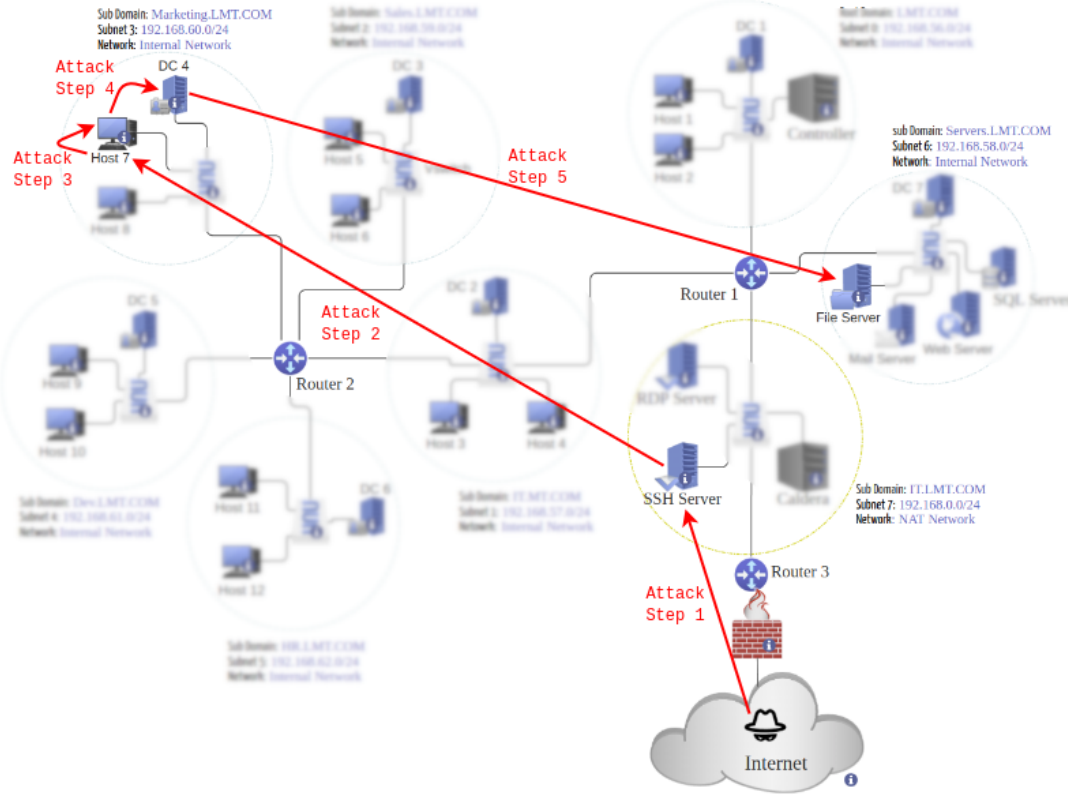


Fig. 3.3.13: Fifth Attack Scenario in LMDG dataset which is a Password Spray attack.

3.3.4.5.5 sixth Attack Scenario In another scenario within the LMDG dataset, a silver ticket attack is demonstrated, which aligns with the Privilege Escalation and Persistence tactics in the MITRE ATT&CK framework. The attack sequence is outlined in Figure 3.3.14. The scenario initiates with an attacker who has access to the SSH server with credentials of the local administrator of subnet 5 (step 1) and then gains access to the DC 6 host within subnet five via SSH (step 2), where the enterprise administrator presented in subnet 0 recently authenticated via SSH.

The attacker begins by downloading the tools that he will need in the attack, like Mimikatz. The attacker starts by dumping the Local Security Authority Subsystem Service (LSASS) memory on the compromised host using Mimikatz. This action exposes recent authentication tokens, including the enterprise administrator's NTLM hash. In this attack, the attacker utilizes Mimikatz to perform a Silver Ticket attack against a specific service within a Kerberos-enabled network.

The attacker forges a service ticket for a service on the target machine using the

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

enterprise administrator’s credentials. The forged ticket is created by specifying the user, domain, service, and the NTLM hash of the administrator’s hash. This service ticket is injected into the current session using the (Pass-the-Ticket) option, allowing the attacker to authenticate as the enterprise Administrator account to access the service without needing a valid Ticket Granting Ticket (TGT) (step 3).

Unlike a Golden Ticket, which grants broad access across the domain, the Silver Ticket is restricted to the targeted service but still allows the attacker to bypass standard authentication mechanisms and gain unauthorized access to the service, potentially enabling the exfiltration of sensitive data or further exploitation of the network.

After performing this attack in step 3, the attacker will now move to step 4 to try to access the forbidden data only accessible by the enterprise administrator on the file server on subnet 6.

The attack was ultimately unsuccessful, likely due to several technical barriers. If detected as a replay attempt, Kerberos replay protection mechanisms may have blocked the reused TGT. Additionally, session and context constraints tied the TGT to specific network contexts, potentially limiting its effectiveness across subdomains. Strict domain policies and auditing features on the Windows Server may have flagged or blocked suspicious ticket requests, especially from unexpected endpoints or high-privilege accounts. Lastly, service ticket signature validation might have prevented unauthorized access if the target server verified ticket integrity. Although the attacker obtained the TGT by step 2, these constraints may have halted further progress, preventing access to restricted resources in subnet 4.

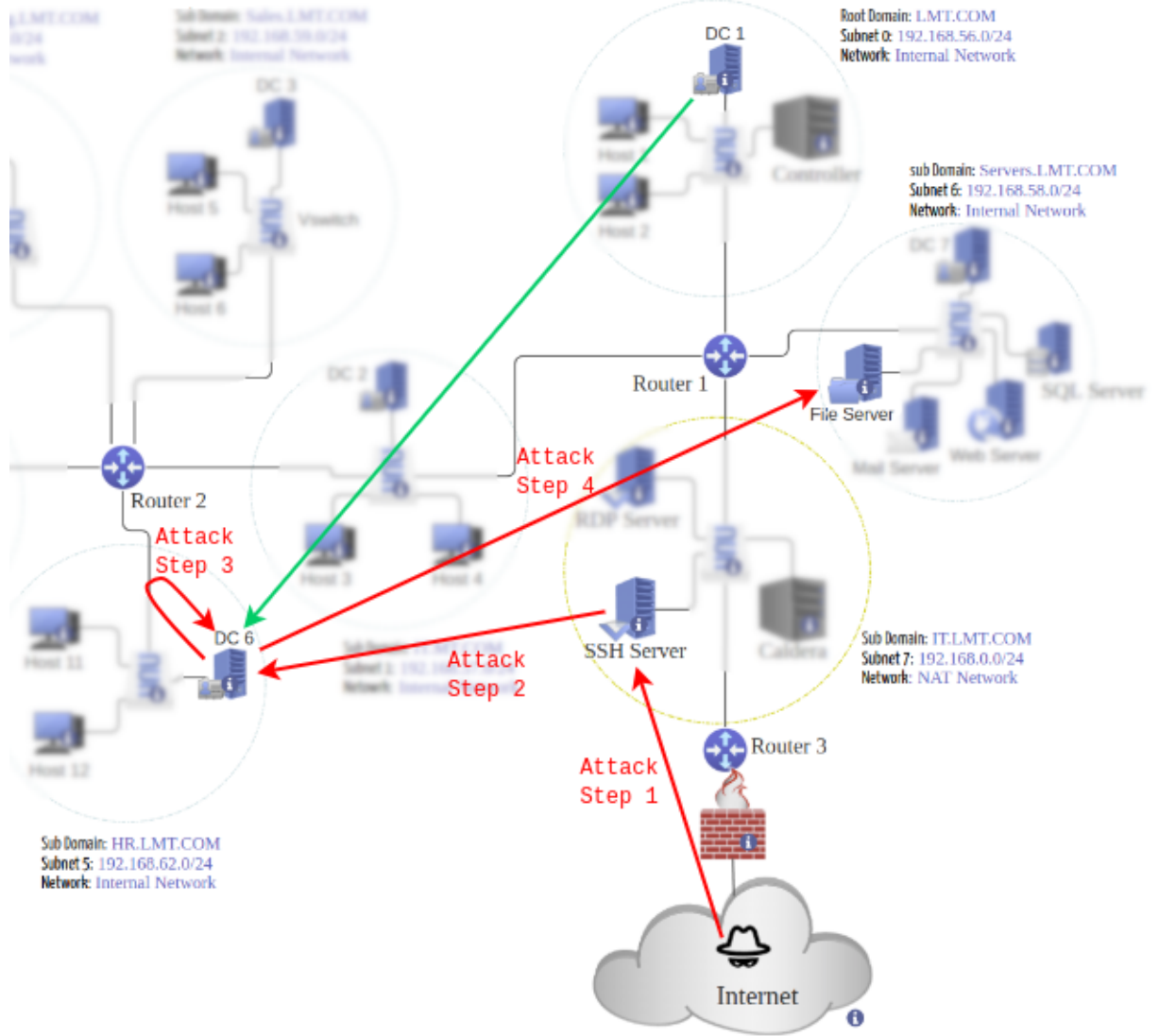


Fig. 3.3.14: Sixth Attack Scenario in LMDG dataset which is a Silver Ticket attack. This figure also illustrates the traversal behavior (hops) in the Golden Ticket attack scenario outlined in Subsection 3.3.4.5.6, providing a step-by-step visualization of the movement through network nodes.

3.3.4.5.6 seventh Attack Scenario In another scenario within the LMDG dataset, a Golden Ticket attack is demonstrated, aligning with the Privilege Escalation and Persistence tactics in the MITRE ATT&CK framework. The attack sequence, as detailed in Figure 3.3.14, initiates with an attacker who initially gains access to the SSH server using the credentials of a local administrator in subnet 5 (step 1) and subsequently gains access to the DC 6 host within subnet five via SSH

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

(step 2), where the enterprise administrator in subnet 0 recently authenticated to the domain controller.

Upon accessing the domain controller, the attacker downloads tools such as Mimikatz. Using Mimikatz, the attacker dumps the Local Security Authority Subsystem Service (LSASS) memory, revealing cached authentication tokens, including the NTLM hash and AES key of the enterprise administrator’s account. Leveraging this information, the attacker generates a Golden Ticket—a forged Kerberos Ticket Granting Ticket (TGT) for the domain administrator account, utilizing the `/aes256` parameter to specify the administrator’s AES key.

The attacker then forges the Golden Ticket with Mimikatz. This crafted ticket allows the attacker to impersonate the enterprise administrator across the entire domain (step 3). Unlike a Silver Ticket, which limits access to a specific service, a Golden Ticket provides domain-wide access to any Kerberos-enabled resource without needing a valid initial TGT, enabling full administrative privileges and sustained persistence within the network. After creating and injecting the Golden Ticket in step 3, the attacker attempts to use it in step 4 to access sensitive files exclusive to the enterprise administrator on a file server within subnet 6.

The attack was ultimately unsuccessful, possibly due to several security constraints. Kerberos replay protection may have flagged frequent Golden Ticket use as a replay attempt, blocking access. Session and context constraints, which bind sessions to specific subdomain contexts, could have limited the ticket’s effectiveness across domains. Additionally, stringent domain policies and auditing on Windows Server might have detected unusual ticket activity, especially high-privilege tickets appearing in new subdomains. Finally, service ticket signature validation on the target server may have rejected unauthorized tickets despite the Golden Ticket’s privilege escalation. These defenses likely hindered further steps, preventing access to the protected file server in subnet six and data exfiltration.

3.3.5 LMDG Labelling Engine (LE)

In cybersecurity datasets, labeling involves identifying and extracting records associated with attack activities from system logs and network traffic. The Labeling Engine serves as the component responsible for automating this extraction process. This subsection will examine the challenges of achieving accurate labeling and introduce our innovative labeling methodology.

3.3.5.1 Challenges in Attack Data Labeling

A thorough review of the literature on labeling techniques for cybersecurity dataset generation reveals that there are primarily three approaches for automatically labeling attack-related records, as outlined in [47]. These approaches are *Injection Timing*, *Behavioral Profiling*, and *Network Security Tools*.

The *Injection Timing* approach labels all system logs or network traffic within a defined time window—precisely, the period in which the attack occurred—as malicious. This technique is frequently employed in research, either on its own or in combination with other labeling methods, to improve labeling accuracy [50, 9, 21, 32, 36]. However, Injection Timing operates on the strong assumption that no benign events or traffic will occur within the attack timeframe; thus, all events within this period are labeled as malicious. This assumption often proves inaccurate, particularly for complex attacks, such as lateral movement, which can be interwoven with benign activity or even exploit benign processes as part of their execution. Consequently, using Injection Timing for labeling in cases of lateral movement or other sophisticated attacks can lead to substantial mislabeling and fails to achieve the necessary accuracy.

The second approach for automatic labeling is the *Behavioral Profiles* method, which relies on predefined behavioral profiles for malicious and benign actions to facilitate labeling. This approach identifies attack-specific characteristics, such as originating from specific machines, allowing records associated with those machines to be easily labeled as malicious. Behavioral profiles thus capture unique attack traits that streamline the labeling process and are widely utilized in the literature

[15, 72, 68]. However, this method proves ineffective in lateral movement attacks, where legitimate hosts and user accounts are leveraged to conduct malicious actions, rendering behavioral profiles insufficient for accurate labeling.

The ***Network Security Tools*** approach for automatic data labeling utilizes information generated by network security tools, including packet sniffers, honeypots, and intrusion detection systems (IDS), to classify records as malicious or benign [2, 68]. This method leverages the detection capabilities of these tools to label network traffic or system events based on observed signatures or anomalous behavior patterns. However, this approach can suffer from significant accuracy limitations due to the inherent weaknesses of the tools, which are prone to generating false positives (incorrectly labeling benign data as malicious) and false negatives (failing to identify actual malicious activity). The tendency of IDSs, for instance, to produce excessive alerts can lead to mislabeling, thus diminishing the reliability of this approach for precise labeling in cybersecurity datasets.

As demonstrated, there is a critical need for a more accurate and precise automatic labeling technique capable of generating high-quality datasets suitable for training machine learning detection models. In response, we introduce a novel and robust labeling methodology designed to address the limitations of existing approaches. Our method significantly improves labeling accuracy, making it particularly well-suited for scenarios involving lateral movement and advanced persistent threats (APTs), where traditional methods often fall short.

3.3.5.2 LMDG Labeling Engine

Our labeling methodology builds upon and extends the labeling approach introduced in [33], with specific enhancements and improvements outlined in the related work section 3.6.1.1.2. We designate this approach as ***process tree labeling***, which can be considered an additional automatic labeling technique and, as we argue, the most accurate among those reviewed. The effectiveness of process tree labeling relies on the client-server architecture introduced in 3.3.4.3 and 3.3.4.4 for automating attack execution, a dependency explored in greater detail in [33].

Upon the completion of attack execution, the LMDG labeling engine, along with its input—a descriptive file containing metadata on attack steps—operates from the controller, depicted in Figure 3.3.1. The engine distributively performs the labeling task across each affected host, using the defined attack steps from the input file to extract the relevant subset of system logs and network connections associated with each attack stage on every affected host. The LMDG labeling engine completes this process in three primary phases: ***Attack Steps Forest Construction***, ***System Logs Labeling***, and ***Network Traffic Labeling***. Before detailing these stages, we will first discuss the input to this engine, namely the descriptive file containing attack steps metadata.

3.3.5.2.1 LMDG Labeling Engine Input The input to the labeling engine consists of a set of hosts impacted by various attack steps, where each host includes a collection of malicious processes with specific attributes. Let H represent the set of all such hosts, i.e., $H = \{h_1, h_2, \dots, h_n\}$, with each host $h_i \in H$ uniquely identified by a `HostName`. For each host h_i , let $P(h_i)$ denote the set of processes associated with the malicious agents deployed on that host during any attack step, i.e., $P(h_i) = \{p_1, p_2, \dots, p_m\}$. Each process $p_j \in P(h_i)$ is described by the following tuple

$$p_j = (\pi, t_s, t_e, \sigma, \nu, \tau, \kappa, \phi)$$

In this tuple, π denotes the process identifier associated with a deployed Caldera agent, i.e., PID. t_s and t_e define the time window during which a particular attack step occurred (start time and end time). The specific step within the attack and the overarching scenario are identified by the κ and σ fields in the tuple, with ϕ indicating whether the step was completed successfully. Since an attack scenario can be executed across various hosts or subnets, multiple versions of the same scenario may exist. For example, in subsection 3.3.4.1, the pass-the-hash (PtH) attack can be executed on

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

different subnets (e.g., subnet 4 instead of subnet 1). This versioning is captured by the ν field in the tuple. Additionally, we may execute the same scenario version multiple times; thus, the τ field is included to distinguish between these instances, offering clear differentiation across repeated executions or trials.

Each host $h_i \in H$ can be formally represented as a tuple containing its **HostName** and the set of associated processes, $P(h_i)$, for each Caldera agent deployed on that host. Formally, this is expressed as $h_i = (\text{HostName}_i, P(h_i))$. The overall input structure can then be denoted by $\text{Input} = \{h \mid h \in H\}$, where each host h belongs to the set H . This structured organization facilitates the grouping of processes by host, enabling efficient distribution of the labeling process and correlation of process executions with the various stages of distinct attack scenarios. For instance, for a host h_1 with **HostName** as `DC01.it.lmt.com`, part of the set of processes associated with it might look like:

$$P(h_1) = \left\{ \begin{array}{l} (4712, 2024-08-04T15:31:28Z, 2024-08-04T17:12:07Z, 1, 2, 1, 2, 1), \\ (1604, 2024-08-09T09:20:18Z, 2024-08-11T13:26:20Z, 3, 1, 1, 5, 0) \end{array} \right\}$$

For each host, the metadata attributes, such as π , t_s , and others, are gathered from Caldera reports generated post-attack execution. Within Caldera’s framework 3.3.4.4, we define an *Adversary* (Figure 3.3.10) for each attack step within a specific scenario. An *Operation* is then executed using this *Adversary*, and it is within the *Operation*’s execution report that all relevant metadata can be retrieved. It is important to note that while Caldera does not natively provide components to define attack steps, versions, or trials explicitly, we achieve this differentiation through our custom naming conventions for *Adversaries* and their associated *Abilities* and the way we use other Caldera components, effectively encoding these distinctions into the metadata collected.

For a specific host h_i , the set of associated malicious processes, $P(h_i)$, may contain

multiple processes, e.g., p_r and p_k , with identical process ID value π within their associated tuples. This duplication arises because the same Caldera agent, identified by its process ID π , can execute multiple attack steps. The other fields within the tuples of p_r and p_k , such as start time t_s , end time t_e , attack step κ , and others, will differ accordingly.

As shown in Figure 3.3.10, which provides an example of an adversary setup, each defined ability (represented as a row) is specified with its corresponding MITRE ATT&CK tactics and techniques, and even a descriptive label may be added. Consequently, in our labeling process, system logs, and network traffic are labeled by the attack step and the relevant tactics and techniques associated with each step, enhancing the granularity and interpretability of the labeling outputs. Next, we will move to the explanation of the three main steps of our labeling methodology.

3.3.5.2.2 Attack Steps Forest Construction The first essential step in our labeling engine is the *Attack Steps Forest Construction*, upon which the subsequent steps rely. In this phase, we construct attack process trees for each host h_i based on the set of processes $P(h_i) = \{p_1, p_2, \dots, p_m\}$, where $P(h_i)$ represents the collection of malicious processes associated with the host during any attack step. For each process $p_j \in P(h_i)$, a process tree is constructed, rooted at the process identifier (PID) of p_j . This tree is further constrained within the start and end times, t_s and t_e , of p_j , ensuring that all descendant processes of p_j 's PID fall within this time interval. This temporal constraint ensures that each process tree captures the causally and temporally relevant events surrounding each attack step, laying the foundation for accurate step-level labeling.

In this step, we construct the forest \mathcal{F} , which is defined as a collection of m distinct trees, each corresponding to a process $p_j \in P(h_i) = \{p_1, p_2, \dots, p_m\}$. Each tree $\mathcal{T}_{p_j} \in \mathcal{F}$ captures the hierarchical structure of all descendant processes initiated by the root process p_j and constrained with the interval $[t_s, t_e]$, effectively representing the malicious process tree for the specific attack step. This structured forest \mathcal{F} therefore serves as a comprehensive representation of attack-related process executions across

different attack steps on host h_i .

Algorithm 3.3.2 Attack Steps Forest Construction

```

1: procedure ATTACKSTEPSFORESTCONSTRUCTION( $H$ )
2:    $\mathcal{F} \leftarrow \emptyset$  ▷ Initialize the forest of attack steps
3:   for  $h_i \in H$  do
4:     for  $p_j \in P(h_i)$  do
5:        $\pi \leftarrow p_j.\pi, \quad t_s \leftarrow p_j.t_s, \quad t_e \leftarrow p_j.t_e$ 
6:        $\sigma \leftarrow p_j.\sigma, \quad \nu \leftarrow p_j.\nu, \quad \tau \leftarrow p_j.\tau$ 
7:        $\kappa \leftarrow p_j.\kappa, \quad \phi \leftarrow p_j.\phi$  ▷ Extract  $p_j$ 's attributes
8:        $\mathcal{L} \leftarrow \emptyset$  ▷ Initialize a list for process IDs
9:        $\mathcal{T}_{p_j} \leftarrow \text{GETPROCESSTREE}(\pi, t_s, t_e, \mathcal{L})$  ▷ Build process tree of process
10:       $p_j$ 
11:       $\mathcal{T}_{p_{j_{\text{meta}}}} \leftarrow (\mathcal{T}_{p_j}, t_s, t_e, \sigma, \nu, \tau, \kappa, \phi)$ 
12:       $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathcal{T}_{p_{j_{\text{meta}}}}\}$ 
13:    end for
14:  return  $\mathcal{F}$ 
15: end procedure
16: procedure GETPROCESSTREE( $\pi, t_s, t_e, \mathcal{L}$ )
17:    $\mathcal{L} \leftarrow \mathcal{L} \cup \{\pi\}$ 
18:    $\mathcal{E} \leftarrow \{e \in \mathcal{E}_{h_i}^{4688} \mid e.\pi = \pi \wedge t_s \leq e.t \leq t_e\}$ 
19:   for  $e \in \mathcal{E}$  do
20:     if  $e.\pi \notin \mathcal{L}$  then
21:        $\mathcal{L} \leftarrow \mathcal{L} \cup \{e.\pi\}$ 
22:       GETPROCESSTREE( $e.\pi, t_s, t_e, \mathcal{L}$ )
23:     end if
24:   end for
25:   return  $\mathcal{L}$ 
26: end procedure

```

The specifics of this step are outlined in Algorithm 3.3.2, where the set $\mathcal{E}_{h_i}^{4688}$ represents all process creation events recorded in the Windows Security log for host h_i . These events correspond precisely to Windows Event ID 4688, which logs each instance of process initiation. More formally, if \mathcal{E}_{h_i} is the set of all Windows events in host h_i then $\mathcal{E}_{h_i}^{4688} = \{e \mid e.\text{EventID} = 4688 \wedge e \in \mathcal{E}_{h_i}\}$

An example of the output generated by Algorithm 3.3.2 is shown in Figure 3.3.15. This output corresponds to the example previously detailed in Subsection 3.3.4.1, which illustrates a Pass-the-Hash (PtH) attack scenario, as depicted in Figure 3.3.7.

In Figure 3.3.15, the example demonstrates two process trees rooted at the same

process ID π of p_r . The first tree is constructed within the constrained time interval $[t_1, t_2]$, encompassing all subprocesses that occurred within this interval and represents attack step 3 from Figure 3.3.7. The second tree is built under the time constraint $[t_3, t_4]$, including all subprocesses within this later interval, and represents attack step 4 from the same figure 3.3.7.

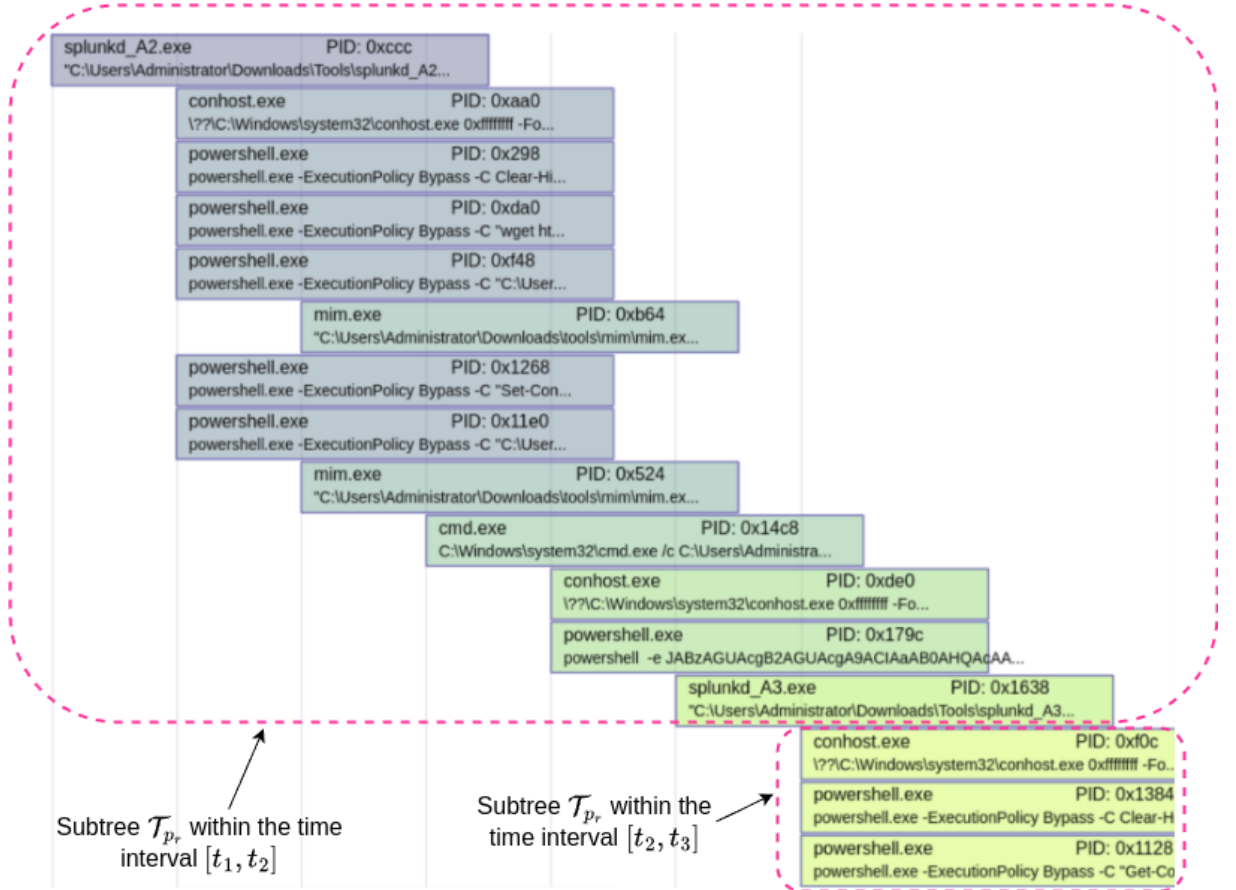


Fig. 3.3.15: Output of algorithm 3.3.2 showing two process trees rooted at the same malicious process p_r . The first tree, representing attack step 3, and the second tree, representing attack step 4 in Figure 3.3.7 explained in subsection 3.3.4.1.

We next move to the subsequent steps, System Logs Labeling and Network Traffic Labeling, which depend on the constructed attack steps forest \mathcal{F} .

3.3.5.2.3 System Logs Labeling In this step, for each host h_i , we iterate over the set \mathcal{L}_{h_i} , which represents the collection of all Windows event logs on that host. For each log $l \in \mathcal{L}_{h_i}$, we further iterate over the trees in the constructed forest \mathcal{F} ,

where each tree $\mathcal{T}_{p_j} \in \mathcal{F}$ corresponds to a specific attack step. The primary objective here is to examine whether the current log l contains any events with process IDs matching those within the current tree \mathcal{T}_{p_j} that occurred within the specified time interval $[t_s, t_e]$ associated with \mathcal{T}_{p_j} . If such events are found, they are extracted and tagged with metadata inherited from \mathcal{T}_{p_j} , including details like the attack scenario, version, step number, and step success status. This labeling process operates at the step level, incorporating relevant MITRE ATT&CK tactics and techniques to contextualize each event within the broader attack framework. The details of this step is shown in algorithm 3.3.3.

Algorithm 3.3.3 Event Log Labeling

```

1:  $\mathcal{E}_{\text{labeled}} \leftarrow \emptyset$  ▷ Initialize labeled events
2: for each  $l \in \mathcal{L}_{h_i}$  do
3:   for each  $\mathcal{T}_{p_j} \in \mathcal{F}$  do
4:      $\pi \leftarrow p_j.\pi$  ▷ Extract tree metadata
5:      $[t_s, t_e] \leftarrow [p_j.t_s, p_j.t_e]$ 
6:      $(\sigma, \nu, \tau, \kappa, \phi) \leftarrow (p_j.\sigma, p_j.\nu, p_j.\tau, p_j.\kappa, p_j.\phi)$ 
7:      $\mathcal{E}_{l,t} \leftarrow \{e \in l \mid e.\pi \in \mathcal{T}_{p_j} \wedge t_s \leq e.t \leq t_e\}$  ▷ Filter events matching  $\pi$  and time interval
8:     if  $\mathcal{E}_{l,t} \neq \emptyset$  then
9:       for each  $e \in \mathcal{E}_{l,t}$  do
10:         $e \leftarrow (e, \sigma, \nu, \tau, \kappa, \phi)$  ▷ Label with tree metadata
11:         $\mathcal{E}_{\text{labeled}} \leftarrow \mathcal{E}_{\text{labeled}} \cup \{e\}$ 
12:      end for
13:    end if
14:  end for
15: end for
16: return  $\mathcal{E}_{\text{labeled}}$ 

```

3.3.5.2.4 Network Traffic Labeling. In this step, we construct the set $\mathcal{E}_{h_i}^{5156} = \{e \mid e.\text{EventID} = 5156 \wedge e \in \mathcal{E}_{h_i}\}$, which represents the collection of Windows events with Event ID 5156, corresponding to the Windows Filtering Platform (WFP). The WFP monitors and filters network traffic on Windows systems, and \mathcal{E}_{h_i} denotes the set of all Windows event logs at host h_i . Subsequently, we iterate over the trees in the constructed forest \mathcal{F} and examine whether any process ID in the current tree \mathcal{T}_{p_j} matches a process ID from the events in $\mathcal{E}_{h_i}^{5156}$. If a match is found, we filter

the relevant events and label them with the metadata of the corresponding tree \mathcal{T}_{p_j} , including attack scenario, version, attack step, step success, and so on. Similar to the System Logs Labeling step, we also consider MITRE ATT&CK tactics and techniques for contextualizing the events within the attack framework.

Algorithm 3.3.4 Network Traffic Labeling

```

1:  $\mathcal{E}_{\text{labeled}} \leftarrow \emptyset$  ▷ Initialize labeled events
2:  $\mathcal{E}_{h_i}^{5156} \leftarrow \{e \mid e.\text{EventID} = 5156 \wedge e \in \mathcal{E}_{h_i}\}$  ▷ Construct the set of WFP events for host  $h_i$ 
3: for each  $\mathcal{T}_{p_j} \in \mathcal{F}$  do
4:    $\pi \leftarrow p_j.\pi$  ▷ Extract tree metadata
5:    $[t_s, t_e] \leftarrow [p_j.t_s, p_j.t_e]$ 
6:    $(\sigma, \nu, \tau, \kappa, \phi) \leftarrow (p_j.\sigma, p_j.\nu, p_j.\tau, p_j.\kappa, p_j.\phi)$ 
7:   for each  $e \in \mathcal{E}_{h_i}^{5156}$  do
8:     if  $e.\pi = \pi$  and  $t_s \leq e.t \leq t_e$  then
9:        $e \leftarrow (e, \sigma, \nu, \tau, \kappa, \phi)$  ▷ Label with tree metadata
10:       $\mathcal{E}_{\text{labeled}} \leftarrow \mathcal{E}_{\text{labeled}} \cup \{e\}$ 
11:    end if
12:  end for
13: end for
14: return  $\mathcal{E}_{\text{labeled}}$ 

```

The process of network flow labeling can be effectively performed using the packet capture (PCAP) files collected from each host, as mentioned in subsection 3.3.2. To associate these flows with specific attack steps, the labeling process leverages the labeled event set $\mathcal{E}_{\text{labeled}}$, which is constructed as described in Algorithm 3.3.4.

Based on the steps and algorithms presented, we argue that our automated labeling methodology, referred to as "Process Tree Labeling," offers superior accuracy compared to other existing automatic labeling techniques. The approach's effectiveness stems from its ability to systematically associate process activities with precise attack steps, leveraging temporal and contextual information from system logs and network traffic data. This method enhances the fidelity of the labeling process, ensuring that each event is accurately attributed to its corresponding stage within the attack lifecycle, thereby improving the overall precision and reliability of the labeling results.

3.4 Dataset

The experimental environment comprises 25 virtual machines (VMs), including a Controller, a Caldera server, domain controllers, application servers, hosts, and routers. Over 22 valid user accounts were set up, but only 11 user credentials were leveraged by the Benign Data Engine to generate benign data on 11 hosts. Windows Event logs and PCAP files were collected from all Windows machines, excluding the Controller; no system logs or PCAPs were collected from the Caldera server. Additionally, PCAP files were captured from routers 1 and 2 to provide supplementary network data, though this traffic is also captured in the PCAP files from the hosts.

The dataset was generated over 25 days, from October 10, 2024, to November 3, 2024. The Benign Data Engine continuously simulated employee behavior throughout this time, producing benign data. Attack executions took place over 10 days, from October 23, 2024, to November 1, 2024, resulting in a dataset containing both benign and malicious activity during these days. The dataset exclusively contains benign data for the initial 14 days before October 23, 2024.

We present some statistics and insights on attacks execution within the LMDG dataset. Figure 3.4.1 illustrates the *Daily Distribution of Attack Steps*, with a histogram depicting the frequency of attack steps executed over time; each bar represents the count of steps occurring on a particular day, with the x-axis denoting individual days and the y-axis showing the number of occurrences, total size of attacks in LMDG dataset is less than 1%.

Figure 3.4.2 displays the *Timeline of Attack Step Occurrences by Scenario*, using a scatter plot to show the timing of attack steps across various days; each point represents an occurrence at a specific day and time, with the x-axis indicating dates and the y-axis showing the time of day to reveal daily distribution patterns. Distinct color coding allows for quick differentiation among scenarios.

Figure 3.4.3 shows the *Frequency Distribution of Scenario and Version Pairs*, where a bar plot represents the count of occurrences for each unique (Scenario, Version) pair, with the x-axis listing scenario-version pairs and the y-axis showing occur-

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

rence counts, facilitating an effective comparison of attack frequencies across different scenario versions. The complete execution timeline is provided in tabular format and is available on our GitHub repository [56].

The total compressed dataset size, encompassing benign and malicious data (excluding router data), is 253 GB; when router data is included, the dataset size increases to 527 GB. Specifically, the compressed PCAP file from router 1 is 201 GB, and that from router 2 is 72 GB. The total uncompressed dataset amounts to 944 GB, with 900.93 GB comprising PCAP files and 43.38 GB for system log files. Additional dataset statistics for the uncompressed data are presented in Table 3.4.1.

The presented numbers reflect the raw dataset characteristics, offering a foundation for extensive feature extraction. Similar to the LANL 2015 dataset, which includes a wealth of authentication data, our dataset allows for detailed extraction of authentication-related features and patterns. These authentication records and additional contextual information provide valuable insights for developing models in areas such as intrusion detection, behavioral analysis, and user activity monitoring. The dataset’s richness in event data and associated metadata establishes a comprehensive base for various cybersecurity research tasks.

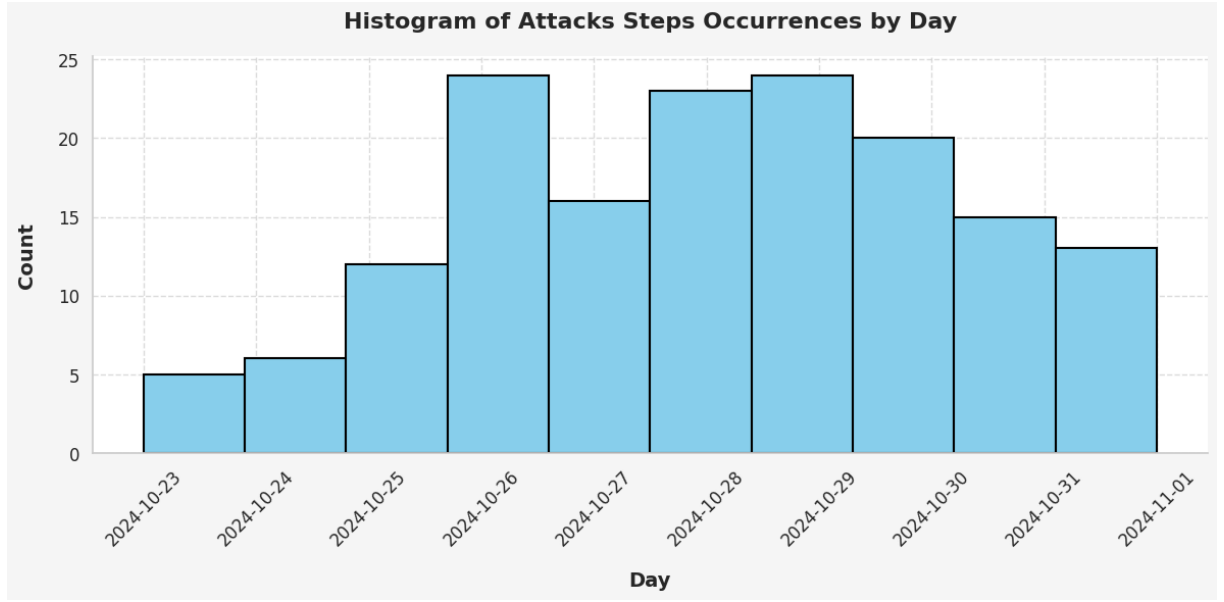


Fig. 3.4.1: **Daily Distribution of Attack Steps:** This histogram visualizes the frequency of attack steps executed over time, with each bar representing the count of attack steps occurring on a specific day. The x-axis denotes individual days, while the y-axis represents the number of occurrences.

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

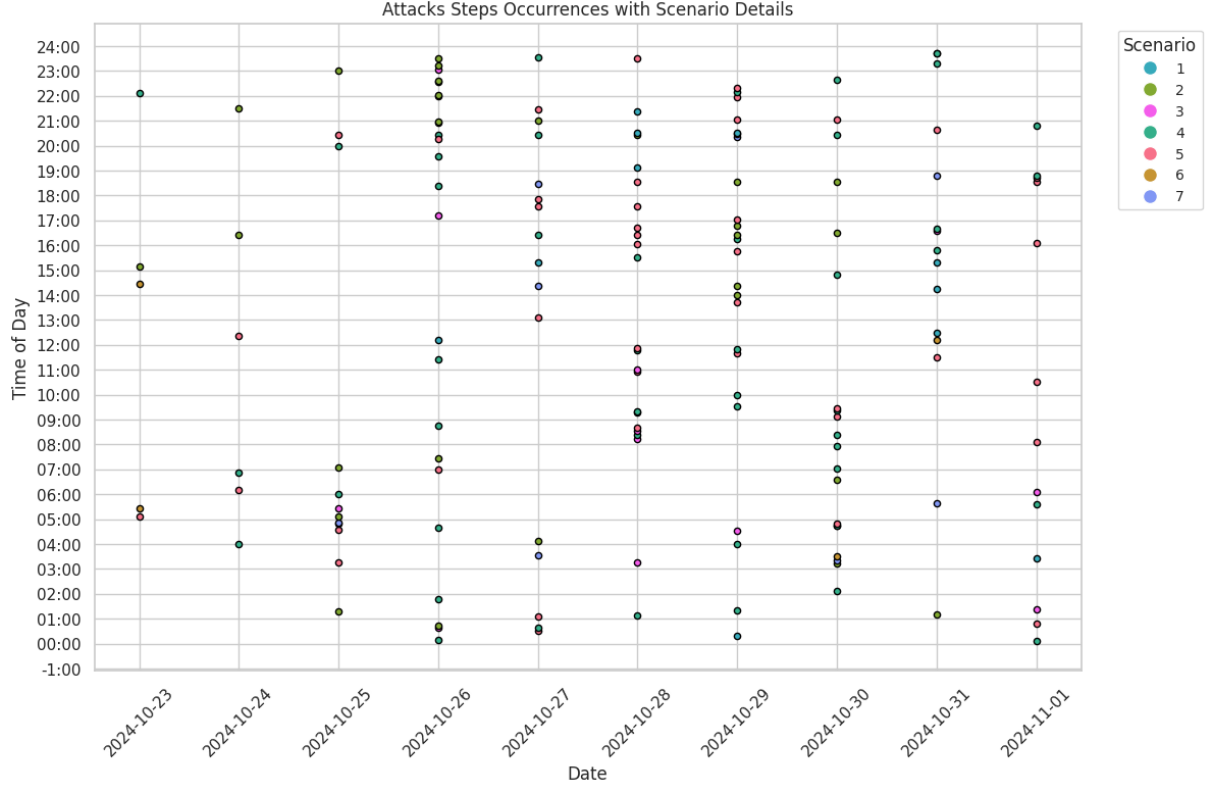


Fig. 3.4.2: **Timeline of Attack Step Occurrences by Scenario:** This scatter plot illustrates the timing of attack steps across various days, with each point representing the occurrence of an attack step on a specific day and time. The x-axis indicates the occurrence dates, while the y-axis represents the time of day to highlight daily distribution patterns. Each scenario is color-coded with a distinct hue, allowing for quick differentiation of scenarios.

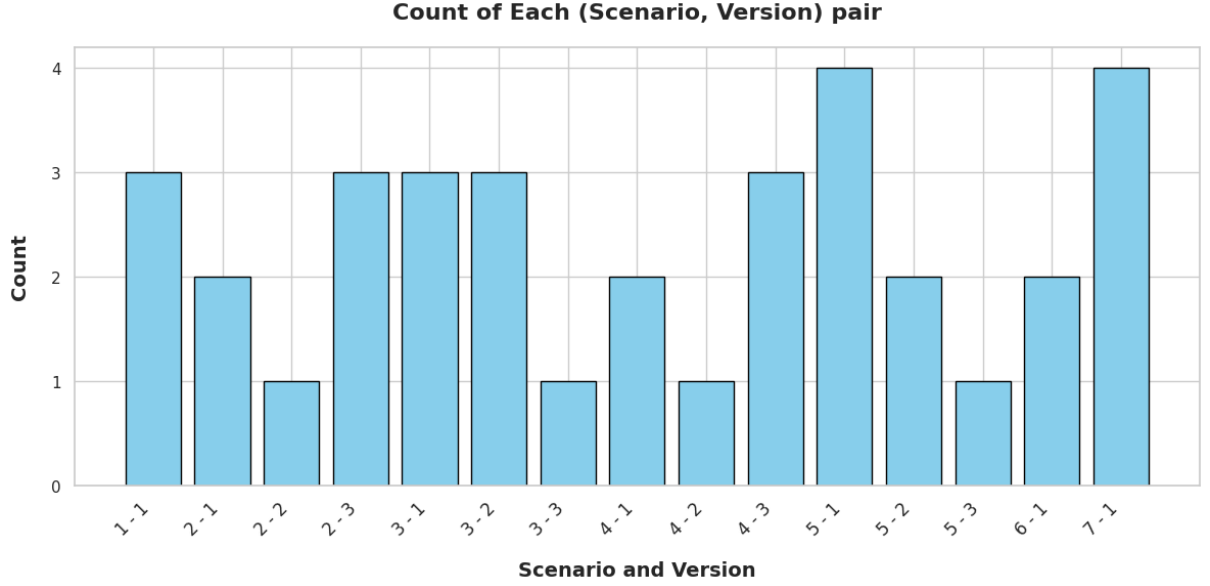


Fig. 3.4.3: **Frequency Distribution of Scenario and Version pairs:** This bar plot displays the count of occurrences for each distinct (Scenario, Version) pair. The x-axis represents individual combinations of scenarios and their respective versions. The y-axis shows the count of occurrences.

Table 3.4.1: Dataset Statistics

Statistic	PCAP Size (GB)	Log Size (GB)
Total Size	900.93	43.38
Average Size	37.54	2.17
Minimum Size	0.51	0.41
Maximum Size	451.00	4.97

3.5 Qualitative Analysis

The quality of a cybersecurity dataset can be regarded as a function of the distinct phases outlined in Figure 3.6.1: specifically, the design of the testbed infrastructure and services, the generation of benign data, the collection of logs and traffic, the execution of attacks, and the labeling process. Each phase plays a critical role in

determining the overall dataset quality, with improvements in the precision and rigor of each phase contributing directly to the robustness and usability of the resulting dataset.

In Section 3.3.2, we introduced the testbed infrastructure, which simulates a realistic enterprise network environment. Within the LMDG framework, virtualization plays a vital role in this phase, enabling the construction of highly realistic and complex testbeds and network architectures. This approach allows for scalable, flexible, and detailed emulation of enterprise networks, enhancing the authenticity and applicability of the dataset for security research.

In the step of benign data generation, or user behavior simulation, we introduced our Benign Data Engine (BDE) in Section 3.3.5.2, as depicted in Figure 3.3.2. The quality of the generated benign data using BDE can be seen as a function of the behavioral scripts provided to it; the more realistic these scripts are, the higher the quality of the resulting data. In our implementation, behavioral scripts were designed based on departmental roles and included a degree of randomization to introduce individual variation within departments. These behaviors simulated activities typical to employees, such as logging in and out, browsing the web, requesting services from local servers, and executing local programs, authentically replicating employee interactions within the network.

For the data collection phase, our dataset—as well as any dataset generated using our framework—is comprehensive, encompassing both system logs and network traffic data, with labeling applied to each. This dual approach ensures the dataset’s completeness, providing a thorough record of activities within the network that allows for detailed analysis of benign and malicious behaviors.

As discussed in Section 3.6.2 on related work, existing datasets of lateral movement (LM) attacks exhibit several limitations, including scarcity of LM instances, outdated attack patterns, limited diversity in techniques, short execution timeframes, and a restricted number of hops. In contrast, the LM attacks executed in our study, detailed in Section 3.3.4.5, address these issues by incorporating numerous LM instances that reflect recent patterns and a variety of techniques. These attacks were executed over

an extended timeframe of 10 days and involved multiple hops across hosts, users, and subnets, with specific scenarios reaching up to 7 hops. Using the CALDERA platform or similar attack emulation tools further enhances flexibility in the design and execution of such attacks, enabling a more realistic and comprehensive dataset.

Section 3.3.5 discusses our automated labeling methodology, specifically *process tree labeling*, and outlines how it achieves a higher level of accuracy compared to other automated labeling methods. This approach is particularly well-suited for labeling lateral movement and advanced persistent threat (APT) attacks, as it allows for precise tracking of process hierarchies and relationships essential in these attack scenarios.

The LMDG framework is designed to support the generation of high-quality datasets through its various integrated components. The LMDG dataset serves as an exemplar of this capability, demonstrating the framework’s effectiveness in producing datasets that are comprehensive, well-structured, and suitable for advanced research and analysis.

3.6 Related Work

This section examines the related work pertinent to our problem from two perspectives. First, 3.6.1, we review existing research on the datasets generation process within intrusion detection and Advanced Persistent Threats (APTs) domains. This review covers the various components of dataset generation, including testbeds infrastructure, dataset collection, benign data generation, attacks execution, and labeling. Particular emphasis is placed on the available frameworks designed to facilitate this process. In the second perspective 3.6.2, we analyze the available datasets in the intrusion detection and APT domains, evaluating the representation of lateral movement (LM) attacks within these datasets. We present our analysis of selected datasets and propose a refined definition of lateral movement based on our findings and conclusions.

3.6.1 Cybersecurity Datasets Generation

In this subsection, we review the available literature on the datasets generation process within intrusion detection and advanced persistent threat (APT) detection, as datasets in these domains can have instances of lateral movement attacks.

An examination of the literature reveals the existence of several frameworks specifically designed for this purpose. By "frameworks," we refer to systems that incorporate varying degrees of automation in the dataset generation process, such as the automation of testbeds, attack simulations, or labeling. In cases where a real network environment is not used, user behavior is typically automated.

As we will discuss, not all frameworks are comprehensive in their collection of both network traffic and system logs. Some focus exclusively on system logs [50], while others are limited to network traffic [22, 9, 43], resulting in incomplete datasets for a holistic analysis.

Our criteria for selecting or filtering frameworks from the literature are based on the framework's ability to handle lateral movement (LM) attacks, particularly regarding attack automation and labeling.

Many frameworks, by design, are not equipped to address lateral movement (LM) attacks [20, 38, 70, 26, 15], as their primary focus is on other attack types, such as scanning, probing, denial-of-service (DoS), or distributed denial-of-service (DDoS) or their labeling techniques are insufficient for accurately handling the complexities of LM attacks. These frameworks are often tailored to detect and label attacks based on specific characteristics, such as network traffic patterns or the identification of attacker IP addresses, which are unsuitable for capturing the complexities and unique behaviors associated with LM attacks 3.1. Consequently, these frameworks fall short in handling the stealthy, multi-stage nature of lateral movement, which typically involves unauthorized internal network traversal and evasion techniques that differ significantly from the more straightforward attack types these frameworks are designed to address.

3.6.1.1 Frameworks

3.6.1.1.1 AIT Framework The AIT framework refers to the notable trilogy of research papers [50, 49, 51] presented by Landauer et al. at the Austrian Institute of Technology (AIT), which we regard as one of the significant contributions for dataset generation. Landauer et al. approached the dataset creation problem from a model-driven engineering perspective [30, 50], which addresses the issue at an abstract level, independent of implementation specifics. This abstraction is crucial for frameworks intended for cybersecurity dataset generation, as it provides the necessary flexibility and scalability expected in such frameworks.

The authors presented a figure (Figure 3.6.1) that encapsulates the critical steps in generating a cybersecurity dataset. To create such a dataset, a testbed must first be established, with the necessary infrastructure deployed. If a real testbed is being used, this step may already be in place. Next, appropriate data collection mechanisms must be implemented to capture benign and malicious behaviors. In the case of synthetic or semi-synthetic dataset generation, A robust user behavior simulation mechanism is also required to generate or define normal user activity within the testbed. Finally, attacks must be executed, and the relevant records associated with these attacks must be accurately labeled. A level of abstraction and automation is required for a framework to be practical for one or more of these steps shown in figure 3.6.1, with particular emphasis on attack automation and labeling—two of the most challenging aspects, as discussed in Sections 3.3.4 and 3.3.5.

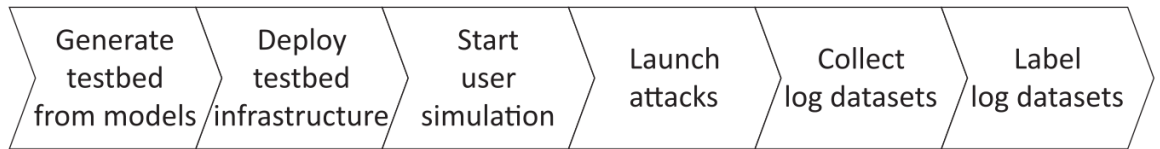


Fig. 3.6.1: Abstracting the problem of cybersecurity datasets generation [51].

We argue that their work offers two primary contributions. The first is their significant advancement in the *automation of testbed infrastructure generation* [50, 49], encompassing steps 1 and 2 in Figure 3.6.1. The second contribution is the

labelling methodology they introduced, which addresses the final step in Figure 3.6.1 [49].

The authors have made significant strides in *testbed infrastructure* automation, leveraging tools like Terraform¹, Ansible², and OpenStack³ to implement their Kyoush platform⁴. This platform is designed to automate the creation of testbeds, allowing researchers to bypass the need to start from scratch for each new use case. Instead, Kyoush enables the reuse of existing components and supports the iterative development of testbeds, making the process more efficient and scalable [50]. This contribution represents one of the few notable advancements in testbed automation, where such tools are crucial for facilitating complex experimentation in cybersecurity research. However, as the authors acknowledge, their approach has some limitations.

A key aspect of Kyoush's design is its model-driven approach, which, while offering flexibility and reusability, introduces additional complexity. Unlike a static testbed that is set up once and used as-is, a model-driven testbed requires the formalization of installation procedures and the separation of fixed and variable components. These components are subject to change as different use cases arise, and the framework must be regularly maintained to ensure its relevance and functionality. This adds to the overall effort required during the initial setup, making the process more resource-intensive than straightforward, static testbeds.

```

1 {
2   "line": 1860,
3   "labels": ["attacker_change_user", "escalate"],
4   "rules": {
5     "attacker_change_user": ["attacker.escalate.audit.su.login"],
6     "escalate": ["attacker.escalate.audit.su.login"]
7   }
8 }
```

Fig. 3.6.2: labelling methodology in AIT.

¹<https://www.terraform.io/>

²<https://www.ansible.com/>

³<https://www.openstack.org/>

⁴<https://github.com/ait-aecid/kyoushi-environment>

The current implementation of the Kyoush platform is also limited in terms of its capabilities. It supports the creation of only three subnets and is restricted to deploying Linux-based virtual machines (VMs). This narrow scope inherently limits the range of testbeds that can be generated, particularly for scenarios that require more diverse environments. For instance, the inability to create Windows VMs constrains the types of user behaviors and attack simulations that can be performed.

Despite these limitations, the authors argue that the increased upfront effort required for model-driven testbed design is offset by the long-term benefits, particularly when testbeds are reused across multiple scenarios. In cases where application requirements evolve, or multiple testbed instances with slight variations are needed, Kyoush’s iterative approach offers considerable advantages. The ability to reuse and adapt testbeds without having to recreate them from scratch not only saves time but also provides greater flexibility in experimental design [50]. While Kyoush presents a promising approach to testbed automation, further development is needed to enhance its flexibility and broaden its applicability.

The second significant contribution of the authors, as outlined in [49], is their **labeling methodology**. This methodology is divided into two primary components. The first component can be characterized as injection timing labeling [47, 35], which operates under the assumption that all traffic or events occurring within a specified time interval are considered malicious. To enhance the accuracy of the generated labels, the authors complement the injection timing technique with an additional approach that evaluates the impact of each attack step on the state of the system logs.

This complementary approach involves crafting tailored queries to extract relevant events from the system logs, thereby allowing for a more precise identification of malicious activities. Through this labeling strategy, the authors successfully pinpoint the logs associated with each specific attack step.

For instance, Figure 3.6.2 illustrates a JSON object that assigns specific labels to an individual log entry within the associated log file. The field "line" in the JSON object indicates the line number of the relevant event in the original log file, while the

"labels" field contains the corresponding classifications. In this case, line 1860 in the log file is labeled with "attacker_change_user" and "escalate," corresponding to the attack step wherein the attacker attains elevated privileges. A detailed examination of this line and other entries in the original log file reveals that these entries correspond to the user authenticating as the root.

However, as the authors acknowledge, two significant limitations are associated with their labeling technique. The first drawback is that this method does not guarantee accurate results and should, therefore, be regarded as a complementary approach to the injection timing labeling method, serving to increase confidence in the labels. This limitation arises because the technique relies on string similarity, i.e., query matching, and thus cannot differentiate between messages that are not sufficiently distinct in the system logs, potentially leading to incorrect labeling [50, 49]. Furthermore, determining the appropriate similarity threshold is challenging, as it depends on the structural characteristics of all potential log events [50, 49].

The second drawback pertains to the manual nature of gathering the expected log entries for each attack step, which involves constructing queries by executing the attacks individually to build the attack dictionary. Any introduction of new attack steps or changes to the logging infrastructure necessitates repeating this labor-intensive process [50, 49]. Consequently, the authors could only perform a single attack in their evaluation [51].

Regarding **attack execution**, the authors designed scripts to automate this process. Still, they did not provide much detail on how the framework handles the automation of more complex attacks, such as lateral movement (LM) attacks 3.3.4. In terms of **dataset collection**, the framework primarily focuses on capturing Linux system logs. The authors developed a User State Machine for **benign data generation** to simulate normal user behavior within the system.

3.6.1.1.2 LADEMU Framework LADEMU, developed by Gjerstad et al. in [33], is a framework designed to generate APT datasets with automatic labeling. Af-

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

ter thoroughly investigating the available frameworks for lateral movement dataset generation, we recognize LADEMU as one of the most promising approaches for this task. The framework introduces a new direction in the accurate labeling of cybersecurity datasets. This novel approach leverages adversary emulation platforms, such as Caldera, to automate attack execution and utilize the information these platforms provide to enhance labeling precision. This contribution marks a significant advancement in dataset generation for cybersecurity research.

Our framework, LMDG, builds upon and advances prior efforts in cybersecurity dataset generation, particularly following a similar approach to that of Gjerstad et al. [33]. However, LMDG introduces several enhancements and improvements, specifically in labeling accuracy and benign data generation, further refining the process to address better the complexities of lateral movement and other sophisticated attack behaviors.

Regarding *testbed infrastructure* generation and associated services, LADEMU operates within a virtualization environment, requiring users to build the testbed using tools such as VirtualBox or VMware. This same approach is employed in our framework, LMDG.

Concerning *dataset collection*, LADEMU primarily captures network data via pcap files and utilizes Sysmon logs for host-based data. Our framework, LMDG, expands upon this by collecting all Windows event logs, not limited to Sysmon (if configured), and performs labeling across the entire set of Windows event logs, enhancing the comprehensiveness of the collected data.

For *benign data generation*, the authors of LADEMU utilize a GHOST tool to automate the generation of realistic user behaviors. In contrast, LMDG introduces a more general approach through its Benign Data Engine (BDE) (section x), which decomposes the problem into two distinct phases: session scheduling for each user and executing behavioral scripts during those sessions. These behavioral scripts can be any script or program, including GHOST, thus offering greater flexibility in simulating user behaviors.

Regarding *attack execution*, LADEMU leverages Caldera with all its capabil-

ities, as discussed in section 3.3.4. This same platform has also been adopted in LMDG to facilitate and automate attack execution.

In terms of **labeling**, LADEMU adopts a similar approach to our method (Section Y), utilizing process IDs of agents to construct malicious process trees corresponding to attack steps. Then, it searches for these processes within the Sysmon log. However, four critical distinctions exist between our approach and LADEMU’s.

Firstly, after constructing malicious process trees—where the root is the Caldera agent—LADEMU considers interactions between processes in these malicious trees and any benign processes. Events associated with benign processes during the interaction window are labeled as malicious, and the interaction typically occurs over a very brief period (in milliseconds), as noted by the authors. This approach presents challenges. To illustrate, consider a scenario in which a process from the malicious tree injects code into a benign process and continues interacting with it over an extended period. The benign process may then carry out both benign and malicious activities, but the malicious effects can manifest long after the interaction [50]. Thus, it is inaccurate to assume that malicious activity only occurs during the interaction window. Moreover, LADEMU labels all events associated with a benign process as malicious if they occur during the interaction interval $[t_1, t_2]$. However, the benign process might simultaneously perform legitimate benign events, leading to mislabeling. To mitigate this issue, LMDG only labels events directly associated with the malicious process trees, avoiding the complexities and inaccuracies of labeling benign processes based on interaction intervals.

Secondly, in constructing malicious process trees, LADEMU relies on the root process ID, which is the Caldera agent, along with Caldera’s start and finish timestamps. While using the start time is a valid approach, as process IDs may repeat over time, relying on the finish time can result in incomplete or “trimmed” trees. As previously discussed, the effects of malicious activities may not manifest immediately and could take an extended period to appear [50]. To address this limitation, LMDG improves upon this by using the root process ID and start time while extending the finish time beyond Caldera’s provided value. This ensures that the process trees are

fully constructed, capturing all relevant activity that might occur after the initial finish time set by Caldera.

Thirdly, LADEMU does not remove the command and control (C&C) signals exchanged between the Caldera server and its agents from the dataset. However, retaining these signals may reduce the realism of the dataset, as in real-world scenarios, C&C traffic may be either encrypted, disguised, or absent in logs available for analysis. To improve the authenticity of the dataset and more accurately simulate real-world environments, LMDG addresses this issue by filtering out C&C signals from the final dataset. This approach ensures that the resulting logs and events are focused on system and network behaviors that would realistically be observed without revealing the underlying orchestration of the attacks, thereby providing a more genuine reflection of an adversarial environment.

Lastly, LMDG introduces a significant enhancement over LADEMU and other existing frameworks by offering the ability to link labeled events with individual attack steps and their corresponding attack scenarios. In complex environments, it is typical for the same attack step to appear across multiple distinct attack scenarios. LMDG enables a more nuanced and context-aware dataset by associating labeled events with their specific scenarios. This added level of detail is crucial for developing and training models aimed at detecting multi-step attacks or constructing advanced threat detection systems. The ability to differentiate between the same attack step occurring in various scenarios aids in understanding the broader attack context. It provides deeper insights into adversarial behaviors, thus improving the fidelity of machine-learning models designed for cybersecurity applications.

3.6.1.1.3 CREME Framework The CREME framework, developed by Bui et al. [15], is designed to generate labeled datasets specifically for training intrusion detection models. In addition to dataset generation, CREME offers a mechanism for evaluating the quality and effectiveness of the datasets it produces. As with other frameworks reviewed, we will analyze CREME’s methodology for each cybersecurity

dataset generation process stage.

Concerning the *testbed infrastructure*, the CREME framework similarly relies on virtualization technologies for constructing its testbeds, a methodology consistent with other frameworks, including LMDG 3.3.2. Additionally, CREME offers a certain level of automation in the initial configuration and service initialization of the testbed once it has been established. This automation reduces manual intervention and ensures that the testbed is properly configured to support subsequent cybersecurity dataset generation and evaluation stages.

Regarding *dataset collection*, the CREME framework captures network traffic in pcap format using tcpdump, collects system logs via rsyslog⁵, and gathers system resource usage information with Atop⁶. However, due to the reliance on rsyslog and Atop, the framework is inherently restricted to Linux and Unix-based environments, limiting its applicability to Windows-based systems.

In terms of *benign data generation*, the framework includes a component called the "Reproduction Module," which handles both the generation of benign data and the execution of attack scripts. This module simulates normal user behavior within the testbed by running benign programs that represent typical activities. However, the authors provide limited details on these scripts' specific nature or contents and do not elaborate on the types of user behaviors simulated through them. This lack of clarity leaves questions regarding the scope and diversity of the behaviors incorporated into the benign data generation process.

Regarding *attack execution*, the "Reproduction Module" is tasked with running the attack scripts and managing various attack phases. The authors conducted five specific attack scenarios: Mirai botnet, ransomware, disk wipe, resource hijacking, and endpoint denial-of-service (DoS) attacks. While these attacks are relatively straightforward and can be automated using pre-defined scripts, the authors do not address more complex attack patterns, such as lateral movement. It remains unclear whether the "Reproduction Module" possesses the maturity or capabilities to man-

⁵<https://www.rsyslog.com/>

⁶<https://linux.die.net/man/1/atop>

age these sophisticated attacks, as platforms like Caldera do effectively 3.3.4. This limitation raises questions about the framework’s extensibility for more advanced adversarial techniques.

The *labeling* approach used in the CREME framework can be categorized as Behavioral Profiles-based [47, 35], where labeling is derived from a prior understanding of the attack programs or compromised machines defined during the configuration phase. For instance, network traffic is labeled malicious if it originates from pre-defined attack machines. The framework assumes that benign programs running on vulnerable clients or injected devices will not access the target server during the attack period. However, as previously discussed 3.3.5, Behavioral Profiles-based labeling is less effective for lateral movement attacks, where both benign and malicious events are generated by the same machine, leading to potential mislabeling.

3.6.2 Available Lateral Movement Datasets

A thorough analysis of publicly available cybersecurity datasets, particularly in intrusion detection, reveals a significant gap: most datasets lack examples of lateral movement (LM) attacks. Prominent datasets such as DARPA, KDD, NSL-KDD, CICIDS2017, and CICIDS2018 exemplify this limitation. Although these datasets have been widely used in research, they focus primarily on other attacks, leaving a critical void for modeling complex, multi-stage attacks like lateral movement.

The existing body of literature offers extensive evaluations and comparisons of these datasets and discussions on the models developed using them. Given the depth of research in this area, we will not reanalyze these datasets here but instead direct readers to relevant references that provide comprehensive overviews [76, 62, 67, 39, 40, 8, 28].

Our study will identify the few datasets that explicitly incorporate LM attacks. We will examine these datasets in detail, analyzing the attack scenarios they contain, the structure of the LM attacks, and the relevance of these datasets for training intrusion detection models capable of detecting multi-stage attacks. This deeper

exploration will help address the critical need for realistic LM data in cybersecurity research, contributing to developing more robust detection systems.

The datasets selected for detailed examination in our study are *LANL* [44], *DARPA Engagement 5* [24], *DARPA OpTC* [29], *PicoDomain* [52], *Pivoting Detection Dataset* [4], *DAPT* [62], *Unraveled* [63].

3.6.2.0.1 LANL Dataset 2015 We will devote particular attention to this dataset, as it is the most widely used for training and evaluating lateral movement detection models. A review of the literature reveals numerous studies that utilize this dataset for such purposes [34, 66, 13, 7, 27, 48, 11, 18].

Two datasets originating from Los Alamos National Laboratory’s corporate (LANL), namely, the ”Unified Host and Network Data Set” [80] and the ”Comprehensive, Multi-Source Cyber-Security Events” (LANL 2015) [44]. The LANL 2018 dataset constitutes a subset of network and host events procured from the LANL enterprise network during an approximately 90-day timeframe; notably, this dataset does not encompass any annotated instances of malicious events, thereby precluding its utility in the evaluation of models for Lateral Movement detection. Conversely, the **LANL 2015** dataset comprises a collection of Windows-based authentication events originating from individual computing nodes and centralized Active Directory domain controller servers spanning a 58-day duration. Additionally, it encapsulates process initiation and termination events sourced from individual Windows-based machines, Domain Name Service (DNS) query activities as observed on internal DNS servers, network flow data originating from various key router locations, and an explicitly delineated array of red teaming exercises designed to exemplify malicious authentication behaviors.

Upon meticulous examination of the malicious authentication incidents, it becomes evident that the manifestation of lateral movement is absent, substantiated by the absence of the pivotal traversal between disparate hosts, as outlined in our proposed definition of lateral movement in section 3.7. To expound further, the malicious

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

data generated by the red team consists of successful authentication events characterized by the following attributes: time, source user, source computer, and destination computer, as illustrated in Table 3.6.2. To analyze potential lateral movement, we construct a directed graph using these attributes. In this graph, an edge is created from the user to the source computer, and another is formed from the source computer to the destination computer. By examining the paths within this directed graph, we aimed to identify lateral movement instances. However, upon analysis, we observed that the longest path in the graph has a length of two, meaning that the sequence "user \rightarrow source computer \rightarrow destination computer" is the longest path present. This suggests that no true lateral movement instances were captured in the dataset. Since the dataset lacks a "destination user" attribute, it is impossible to observe any user switching, as confirmed by our analysis. We can omit the user attribute for visualization purposes and focus on representing the graph solely with source and destination computers. In this simplified directed graph, the longest path is of length one, indicating direct connections between source and destination computers. Figure 3.6.4 illustrates this directed graph, which includes all malicious events identified in the dataset.

Table 3.6.1: Examples of Malicious Authentications in LANL 2015 Dataset

Time	Source User	Source Computer	Destination Computer
151036	U748@DOM1	C17693	C305
151648	U748@DOM1	C17693	C728
151993	U6115@DOM1	C17693	C1173

3.6.2.0.2 LANL Dataset 2015 We will devote particular attention to this dataset, as it is the most widely used for training and evaluating lateral movement detection models. A review of the literature reveals numerous studies that utilize this dataset for such purposes [34, 66, 13, 7, 27, 48, 11, 18].

Two datasets originating from Los Alamos National Laboratory’s corporate (LANL), namely, the ”Unified Host and Network Data Set” [80] and the ”Comprehensive, Multi-Source Cyber-Security Events” (LANL 2015) [44]. The LANL 2018 dataset constitutes a subset of network and host events procured from the LANL enterprise network during an approximately 90-day timeframe; notably, this dataset does not encompass any annotated instances of malicious events, thereby precluding its utility in the evaluation of models for Lateral Movement detection. Conversely, the **LANL 2015** dataset comprises a collection of Windows-based authentication events originating from individual computing nodes and centralized Active Directory domain controller servers spanning a 58-day duration. Additionally, it encapsulates process initiation and termination events sourced from individual Windows-based machines, Domain Name Service (DNS) query activities as observed on internal DNS servers, network flow data originating from various key router locations, and an explicitly delineated array of red teaming exercises designed to exemplify malicious authentication behaviors.

Upon meticulous examination of the malicious authentication incidents, it becomes evident that the manifestation of lateral movement is absent, substantiated by the absence of the pivotal traversal between disparate hosts, as outlined in our pro-

posed definition of lateral movement in section 3.7. To expound further, the malicious data generated by the red team consists of successful authentication events characterized by the following attributes: time, source user, source computer, and destination computer, as illustrated in Table 3.6.2. To analyze potential lateral movement, we construct a directed graph using these attributes. In this graph, an edge is created from the user to the source computer, and another is formed from the source computer to the destination computer. By examining the paths within this directed graph, we aimed to identify lateral movement instances. However, upon analysis, we observed that the longest path in the graph has a length of two, meaning that the sequence "user \rightarrow source computer \rightarrow destination computer" is the longest path present. This suggests that no true lateral movement instances were captured in the dataset. Since the dataset lacks a "destination user" attribute, it is impossible to observe any user switching, as confirmed by our analysis. We can omit the user attribute for visualization purposes and focus on representing the graph solely with source and destination computers. In this simplified directed graph, the longest path is of length one, indicating direct connections between source and destination computers. Figure 3.6.4 illustrates this directed graph, which includes all malicious events identified in the dataset.

3. LMDG: A FRAMEWORK FOR LATERAL MOVEMENT DATASETS GENERATION

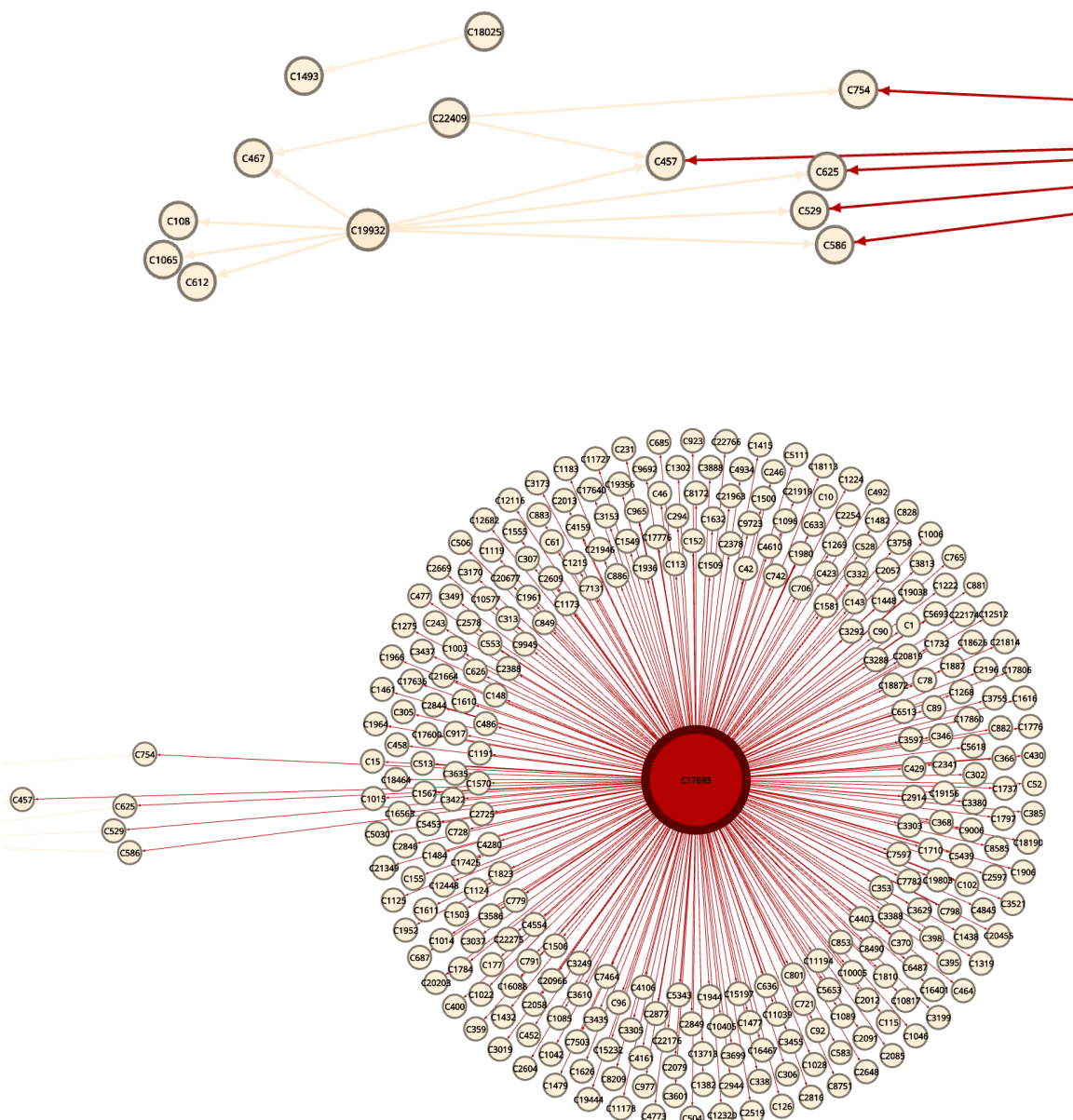


Fig. 3.6.4: LANL 2015 malicious authentications as directed graph.

Table 3.6.2: Examples of Malicious Authentications in LANL 2015 Dataset

Time	Source User	Source Computer	Destination Computer
151036	U748@DOM1	C17693	C305
151648	U748@DOM1	C17693	C728
151993	U6115@DOM1	C17693	C1173

3.6.2.0.3 *DARPA Transparent Computing Engagement 5 (DARPA 2019)*

The configuration within this dataset closely resembled that of Engagement 3 [23], albeit encompassing a larger group of hosts. The assembly consisted of 16 distinct hosts that operated on diverse operating systems, namely Windows, Ubuntu, and Android, mirroring the compositional framework of the preceding dataset. Before and during the engagement, There was a phase of benign data generation. All instances of attack materialized exclusively between 9 a.m. and 5 p.m. on weekdays across eight days. In contradistinction to the third engagement, which doesn't contain lateral movement attacks, the present one comprises two lateral movement scenarios. The first scenario is characterized by a sequence wherein attackers successfully compromise a host within the targeted network, configuring it to function as their command and control hub. After this, they pivoted to another Linux-based host using stolen authentication credentials. The second scenario closely parallels the first one, involving a similar strategy wherein attackers initially gained a foothold on the network, subsequently pivoting onto multiple intermediary hosts through SSH and stolen credentials. Within this dataset, these two instances are the exclusive manifestations of lateral movement. These instances diverge from the protracted temporal characteristics commonly associated with lateral movements as they transpire over a short interval. Both instances share a commonality in their approach, employing an identical technique to accomplish lateral maneuvering.

3.6.2.0.4 *DARPA Operationally Transparent Cyber 2019 (OpTC)* Compared to Engagement 3 [23] and 5 [24], this dataset has many hosts, a thousand hosts in a Windows network, and the data from five hundred hosts was collected rather than from the complete set of hosts due to space constraints. The evaluation started with benign record generation, followed by the red team attacks, which were performed in three days. Benign traffic ran continuously during red team activity. Kafka, an open-source stream-processing server, facilitates information sharing among system components. Windows 10 endpoints employ sensors to monitor host events, packaging them into JSON records sent to Kafka. These records are then translated into eCAR format by a server and reinserted into Kafka. A data analytics component further processes the eCAR records, converting them into a graph structure for analysis and visualization. Within this dataset, two occurrences of lateral movement are identifiable. The initial incident occurred on the first day, involving a sequence of four intermediary transitions across five distinct hosts, with one of these hosts designated the domain controller. The attacker employed Windows Management Instrumentation (WMI) to effectuate the traversal between hosts, augmenting the process by integrating additional techniques. The subsequent occurrence unfolded the next day, likewise leveraging WMI; however, it exhibited greater complexity than its predecessor, characterized by a larger number of intermediary transitions. Similar to the circumstances in Engagement 5, this dataset exhibits a limited number of instances of lateral movement occurring within a concise timeframe, underscored by a deficiency in the array of strategies employed for accomplishment.

3.6.2.0.5 *PicoDomain 2020* The PicoDomain [52] simulation comprised a compact Windows office setting encompassing five workstations, a domain controller, and a gateway firewall/router. This setup is connected to a limited-scale internet housing websites and adversary infrastructure. The internal network featured a Windows Active Directory environment with distinct Organizational Units (OUs): HR, R&D, and a confidential supersecret OU. Scripts mimicking web browsing and SMB file

sharing were utilized, and data collection spanned three days. The Mandiant Attack Lifecycle (MAL) [19] was the framework for outlining the adversary’s campaign strategy, mainly focusing on the recurring phases of hostile campaigns. The initial MAL stages include Initial Reconnaissance, Initial Compromise, and Establish Foothold. Subsequent phases, repeated as necessary, encompass Escalate Privileges, Internal Reconnaissance, Move Laterally, and Maintain Presence. The MAL concludes with the adversary accomplishing their mission. The dataset reveals prevalent attacker engagement in multiple stages. Notably, lateral movement (LM) predominantly utilized WMI and DCOM techniques with minimal diversity. The dataset comprises a single LM scenario over a short three-day span lacking instances with an extended temporal scope.

3.6.2.0.6 *Pivoting Detection Dataset 2017* Within this dataset [3], network traffic information takes shape as network flows observed within a large organization throughout a day. These flows embody the interactions of internal hosts within the observed network setting. Each flow sample in the dataset carries a binary label, indicating its involvement in a pivoting activity. The labeling process underwent manual execution and validation. The dataset’s size reaches about 6 GB, encompassing close to 75 million network flows. The dataset predominantly records pivoting activities where the ”pivoter” host controlled the ”terminal” host remotely using third-party tools like the Windows Remote Desktop protocol. It’s important to note that these actions are typical, non-malicious pivoting activities that occurred within the monitored organization. As such, they depict infrequent and harmless occurrences. As previously noted, this dataset was gathered within a brief timeframe of just one day. This limited duration may not be an ideal representation of Lateral Movement, which often occurs over an extended period. Upon creating a graph that visualizes all the pivoting activities with their temporal progression within the dataset it becomes evident that there are only a few instances of Lateral Movement, all of which involve two hops. The specific technique employed in these pivoting instances remains unclear.

3.6.2.0.7 DAPT 2020 The DAPT2020 [62] dataset was developed as an Advanced Persistent Threat (APT) dataset with two primary objectives: to make attacks indistinguishable from normal traffic and to include traffic across both the public-to-private interface and the internal (private) network. The testbed used was minimal, consisting of two virtual machines—one connected to a private network and the other to a public network—along with a log server and a gateway router. This simplified architecture represents a limitation, as it does not accurately reflect the complexity of real-world environments. Data collection spanned five days, with the first day capturing only normal traffic and the subsequent four days containing various stages of an APT attack. On the fourth day, the lateral movement phase occurred, involving reconnaissance and exploitation activities from the compromised public VM to gain access to critical systems on the internal network. This phase employed tools and techniques such as Nmap for network scanning, the vsftpd 2.3.4 vulnerability, weak SSH authentication, a MySQL script for CVE-2012-2122, and Metasploit. However, the dataset includes only a lateral movement instance executed over a 10-hour window—a significantly shorter timeframe than realistic LM attacks, which can span weeks or even months.

3.6.2.0.8 Unraveled Dataset 2023 The Unraveled dataset [63] builds upon the DAPT2020 dataset, introducing significant enhancements. The testbed architecture has been substantially improved, emulating a realistic enterprise network environment. The system architecture separates corporate and production networks with a firewall. The organization has 15 employees, using Snort as a Network Intrusion Detection System (NIDS) monitored by a Blue Team. The corporate network contains three subnets, each simulating a department with different operating systems. Logs are sent to a centralized ELK server in the production network. The production network consists of a public subnet with a web server and a honeypot and a private

subnet hosting critical services like a database, internal application, and mail server. A firewall regulates traffic, allowing only specific public-to-private connections, while private servers can access the public network freely. An additional enhancement in the Unraveled dataset was the extended execution of the APT attack over six weeks. However, the lateral movement phase remained relatively simplistic, occurring within a single day and consisting of internal reconnaissance and password cracking. In both the DAPT2020 and Unraveled datasets, the attack execution and labeling processes were conducted manually.

Through a deep and thorough qualitative analysis based on the properties of lateral movement attacks discussed in the introduction section 3.1, several recurring limitations across these datasets and others have been identified, which we elaborate on below:

(1) *Scarcity of Lateral Movement Instances:*

A prevalent issue across the examined datasets is the limited availability of lateral movement (LM) attack instances. This scarcity hampers the ability to comprehensively model and detect multi-stage attacks, as datasets must include numerous instances to reflect the diversity and complexity of real-world attack scenarios.

(2) *Outdated Attack Patterns:* Most datasets analyzed need to be updated, capturing older attack strategies that may no longer align with contemporary threat landscapes. In an ever-evolving field, datasets must reflect recent attack patterns to ensure that intrusion detection models remain relevant and effective against emerging threats.

(3) *Lack of Diversity in LM Techniques:* The datasets offer limited variation in the techniques used to perform lateral movement. In practice, attackers employ many methods, including credential theft, remote service exploitation, and process injection. Capturing this diversity is essential for developing robust detection models that generalize well across different LM techniques.

(4) *Short Execution Timeframes:* Many LM attacks in these datasets are

executed over brief periods, failing to capture the prolonged nature of real-world campaigns. LM activities often extend over hours or days, requiring datasets to reflect these extended timelines to allow accurate modeling of such behaviors.

(5) **Limited Number of Hops:** Several datasets restrict LM activities to a minimal number of hops, often just a few connections between systems. However, realistic LM scenarios typically involve multiple hops across various hosts, user accounts, and subnets, better mimicking real-world attack paths.

(6) **Incomplete Data Sources:** Some datasets provide only partial data, such as focusing solely on network traffic or authentication logs. For a dataset to be truly comprehensive, it must integrate multiple data sources, including network flow data and system logs, to enable cross-layer correlation and accurate attack reconstruction.

(7) **Insufficient Emphasis on Labeling Methodology:** An additional challenge lies in clearer methodologies for accurately labeling attack-related events within system and network logs. Proper labeling is critical to associate events with specific attack steps and phases, ensuring that the dataset remains reliable for training and evaluation purposes.

(8) **Simple Testbed Architectures:** Many datasets rely on simplistic testbed architectures, limiting their ability to simulate realistic enterprise environments. To improve the quality and applicability of datasets, it is crucial to develop testbeds that reflect real-world complexities, including multiple network segments, diverse user behaviors, and complex configurations.

It is worth mentioning that despite the growing shift towards cloud-based infrastructure, none of the reviewed datasets provide scenarios involving lateral movement in cloud environments. The absence of such datasets creates a significant gap, as cloud platforms introduce unique attack vectors and challenges that must be studied to enhance cloud security practices.

To summarize, a dataset must include sufficient LM attack instances reflecting recent patterns and diverse techniques to effectively support the training and evaluation of lateral movement (LM), APT, or Multi-step attack detection models. The attacks should span extended timeframes and involve multiple hops across hosts, users, and

subnets. A comprehensive dataset requires system logs and network flows, accurate labeling methods, and generation from realistic testbeds to ensure reliability.

3.7 Discussion

Our review of lateral movement detection reveals a need for a comprehensive definition of Lateral Movement. While MITRE ATT&CK [59] defines it broadly as techniques for accessing and controlling remote systems, this vague description limits the effectiveness of detection models, highlighting the need for a more precise and actionable definition.

We define two key types of adversary progression: *horizontal progression* and *vertical progression*. *Horizontal progression* involves gaining independent access to multiple hosts without interdependence, which does not qualify as lateral movement. In contrast, *vertical progression* describes interconnected access, where controlling one system enables access to others. ***We define lateral movement as vertical progression across hosts, accounts, or privileges, where one access leads to another.*** This includes movement between hosts, accounts with elevated privileges, and privilege escalation. This refined definition is essential for creating effective detection models.

In the cloud environment, lateral movement follows a similar concept with modifications. *Identities* (user, application, and service accounts) correspond to *accounts*, requiring authentication to access resources, while *permissions* or *policies* align with *privileges*, defining access levels. A key distinction in the cloud is the *services layer*, which includes resources like AWS EC2 and S3, providing computing and storage. Thus, *cloud lateral movement involves vertical progression across identities, permissions/policies, services, and resources*. For example, an attack detailed by Microsoft Threat Intelligence [42] involved exploiting SQL injection to access an Azure database server and using the Instance Metadata Service (IMDS) to obtain further access to cloud resources.

Regarding the threat model in the LMDG dataset, it emulates realistic APT-like scenarios where adversaries perform stealthy, persistent attacks over an extended period. These scenarios include initial access, privilege escalation, and multi-hop lateral movement across hosts and network subnets, reflecting the sophisticated behaviors of modern attackers targeting enterprise networks. By leveraging the CALDERA platform for attack emulation, the framework enables flexible attack design, supporting a variety of lateral movement techniques and bypassing typical security defenses. To enhance realism, these attacks occur within a backdrop of benign user activities generated by the Benign Data Engine (BDE), providing a nuanced environment for distinguishing between normal and malicious behavior. This threat model thus offers a robust foundation for evaluating detection and response mechanisms against complex and dynamic cyber threats.

3.8 Conclusions and Future Work

In this work, we have comprehensively examined current cybersecurity benchmark datasets with a specific focus on evaluating the presence and characteristics of lateral movement (LM) attacks. Our analysis, the first of its kind, assessed LM datasets across multiple dimensions, including the quantity and variety of LM techniques, attack duration, number of movement hops, data sources (e.g., authentication logs, network flows), labeling methodologies, and testbed configurations. This investigation has highlighted gaps and challenges within existing datasets, providing insight into the strengths and limitations of current approaches to lateral movement detection.

We developed a benchmark dataset focused explicitly on lateral movement attacks to address the identified limitations. This dataset, designed to overcome many existing issues in LM datasets, provides a valuable resource for the research community, facilitating the training and evaluation of more effective LM detection models. Our qualitative dataset analysis demonstrates its applicability for various lateral movement scenarios. It ensures that the diversity and complexity of attacks are suitable for testing advanced detection techniques.

Additionally, we introduced the Lateral Movement Dataset Generator (LMDG) framework, a reproducible toolset for generating high-quality LM and APT datasets. The LMDG framework automates benign data generation, attack execution, and—crucially—the labeling of attack-related events in system and network logs. Recognizing the challenges posed by automatic labeling in LM scenarios, where benign hosts may perform malicious actions, we proposed a novel technique, *process tree labeling*. This method offers improved precision and accuracy over existing techniques such as injection timing, behavior profiles, and network security tools. Overall, the contributions of this work enhance the landscape of LM dataset generation and analysis, supporting further advancements in cybersecurity research and LM detection capabilities.

Several limitations of our framework warrant consideration. First, using virtualization to construct testbeds necessitates extensive domain expertise, making the process time-consuming and highly case-dependent, as discussed in more detail in [50]. This requirement for specialized knowledge may hinder the framework’s scalability and accessibility. Second, the client-server architecture employed in attack automation, as outlined in Sections 3.3.4.3 and 3.3.4.4, introduces traffic and log accuracy challenges. Specifically, the traffic generated by client-server communication must be filtered to avoid contaminating the dataset with automation-related signals, ensuring that the resulting data remains realistic and reflective of actual attack behaviors. Finally, our proposed labeling methodology, process tree labeling, the most accurate automatic labeling technique, is inherently tied to the client-server automation model. This dependency arises from the need to identify the process IDs of deployed agents, creating a coupling between labeling and attack automation. This coupling is discussed in more detail in [33] and may limit the applicability of our labeling approach in environments where such client-server structures are not feasible.

While this study includes a qualitative analysis of our dataset 3.5 and comparisons with existing datasets in the literature, further work is needed to incorporate quantitative analysis methods. A systematic review of current quantitative assessment techniques used in cybersecurity datasets will enable us to apply rigorous, data-

driven evaluation metrics to our dataset, enhancing its reliability and usability. In addition, future efforts may focus on producing a more comprehensive dataset that encompasses the full spectrum of Advanced Persistent Threat (APT) attack stages rather than concentrating solely on lateral movement. Such a dataset would capture all phases of APT attacks, offering a richer resource for developing and benchmarking holistic detection models that address the complete lifecycle of sophisticated attack vectors. This extension will advance research into multi-stage threat detection, providing excellent value for the cybersecurity community.

References

- [1] Adel Alshamrani et al. “A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2019), pp. 1851–1877. DOI: 10.1109/COMST.2019.2891891.
- [2] Francisco J Aparicio-Navarro, Konstantinos G Kyriakopoulos, and David J Parish. “Automatic dataset labelling and feature selection for intrusion detection systems”. In: *2014 IEEE Military Communications Conference*. IEEE. 2014, pp. 46–51.
- [3] Giovanni Apruzzese et al. “Detection and Threat Prioritization of Pivoting Attacks in Large Networks”. In: *IEEE Transactions on Emerging Topics in Computing* 8.2 (2020), pp. 404–415. DOI: 10.1109/TETC.2017.2764885.
- [4] Giovanni Apruzzese et al. “Detection and Threat Prioritization of Pivoting Attacks in Large Networks”. In: *IEEE Transactions on Emerging Topics in Computing* 8 (2020), pp. 404–415. URL: <https://api.semanticscholar.org/CorpusID:64482369>.
- [5] Lora Aroyo et al. “Data excellence for AI: why should you care?” In: *Interactions* 29.2 (Feb. 2022), pp. 66–69. ISSN: 1072-5520. DOI: 10.1145/3517337. URL: <https://doi.org/10.1145/3517337>.

- [6] Tim Bai et al. “RDP-based Lateral Movement detection using Machine Learning”. In: *Computer Communications* 165 (2021), pp. 9–19. ISSN: 0140-3664. DOI: <https://doi.org/10.1016/j.comcom.2020.10.013>. URL: <https://www.sciencedirect.com/science/article/pii/S0140366420319617>.
- [7] Tim Bai et al. “Rdp-based lateral movement detection using machine learning”. In: *Computer communications* 165 (2021), pp. 9–19.
- [8] Kousik Barik et al. “Cybersecurity deep: approaches, attacks dataset, and comparative study”. In: *Applied Artificial Intelligence* 36.1 (2022), p. 2055399.
- [9] Monowar H Bhuyan, Dhruva K Bhattacharyya, and Jugal K Kalita. “Towards Generating Real-life Datasets for Network Intrusion Detection.” In: *Int. J. Netw. Secur.* 17.6 (2015), pp. 683–701.
- [10] Haibo Bian et al. “Uncovering Lateral Movement Using Authentication Logs”. In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 1049–1063. DOI: 10.1109/TNSM.2021.3054356.
- [11] Haibo Bian et al. “Uncovering lateral movement using authentication logs”. In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 1049–1063.
- [12] Atul Bohara et al. “An Unsupervised Multi-Detector Approach for Identifying Malicious Lateral Movement”. In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. 2017, pp. 224–233. DOI: 10.1109/SRDS.2017.31.
- [13] Benjamin Bowman et al. “Detecting lateral movement in enterprise computer networks with unsupervised graph {AI}”. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 2020, pp. 257–268.
- [14] Benjamin Bowman et al. “Detecting Lateral Movement in Enterprise Computer Networks with Unsupervised Graph AI”. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. San Sebastian: USENIX Association, Oct. 2020, pp. 257–268. ISBN: 978-1-939133-18-2. URL: <https://www.usenix.org/conference/raid2020/presentation/bowman>.

- [15] Huu-Khoi Bui et al. “CREME: A toolchain of automatic dataset collection for machine learning in intrusion detection”. In: *Journal of Network and Computer Applications* 193 (2021), p. 103212.
- [16] José Camacho et al. “Quality In / Quality Out: Data quality more relevant than model choice in anomaly detection with the UGR’16”. In: *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. 2023, pp. 1–5. DOI: 10.1109/NOMS56928.2023.10154333.
- [17] Gustavo de Carvalho Bertoli et al. “Bridging the gap to real-world for network intrusion detection systems with data-centric approach”. In: *CoRR* abs/2110.13655 (2021). arXiv: 2110.13655. URL: <https://arxiv.org/abs/2110.13655>.
- [18] 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)*. IEEE. 2018, pp. 708–715.
- [19] Mandiant (A FireEye Company). “Targeted Attack Lifecycle”. In: (2023). URL: <https://www.mandiant.com/resources/insights/targeted-attack-lifecycle>.
- [20] Carlos Garcia Cordero et al. “ID2T: A DIY dataset creation toolkit for intrusion detection systems”. In: *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2015, pp. 739–740.
- [21] Gideon Creech and Jiankun Hu. “Generation of a new IDS test dataset: Time to retire the KDD collection”. In: *2013 IEEE wireless communications and networking conference (WCNC)*. IEEE. 2013, pp. 4487–4492.
- [22] Robertas Damasevicius et al. “LITNET-2020: An annotated real-world network flow dataset for network intrusion detection”. In: *Electronics* 9.5 (2020), p. 800.

- [23] DARPA I2O. *Transparent Computing Engagement 3 (E3) Dataset*. <https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md>. Accessed: 2024-10-04. 2018.
- [24] DARPA I2O. *Transparent Computing Engagement 5 (E5) Dataset*. <https://github.com/darpa-i2o/Transparent-Computing/blob/master/README.md>. Accessed: 2024-10-04. 2019.
- [25] Remi Denton et al. “Bringing the people back in: Contesting benchmark machine learning datasets”. In: *arXiv preprint arXiv:2007.07399* (2020).
- [26] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D Jurcut. “InSDN: A novel SDN intrusion dataset”. In: *IEEE access* 8 (2020), pp. 165263–165284.
- [27] Yong Fang et al. “LMTracker: Lateral movement path detection based on heterogeneous graph embedding”. In: *Neurocomputing* 474 (2022), pp. 37–47.
- [28] Mohamed Amine Ferrag et al. “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study”. In: *Journal of Information Security and Applications* 50 (2020), p. 102419.
- [29] FiveDirections. *DARPA Operationally Transparent Cyber (OpTC) Dataset*. <https://github.com/FiveDirections/OpTC-data>. Accessed: 2024-10-04. 2019.
- [30] Fermin Galan et al. “Using a model-driven architecture for technology-independent scenario configuration in networking testbeds”. In: *IEEE Communications Magazine* 48.12 (2010), pp. 132–141. DOI: 10.1109/MCOM.2010.5673083.
- [31] Chenquan Gan et al. “Advanced Persistent Threats and Their Defense Methods in Industrial Internet of Things: A Survey”. In: *Mathematics* 11.14 (2023). ISSN: 2227-7390. DOI: 10.3390/math11143115. URL: <https://www.mdpi.com/2227-7390/11/14/3115>.
- [32] Sebastian Garcia et al. “An empirical comparison of botnet detection methods”. In: *computers & security* 45 (2014), pp. 100–123.

- [33] Julie Gjerstad et al. “LADEMU: a modular & continuous approach for generating labelled APT datasets from emulations”. In: *2022 IEEE International Conference on Big Data (Big Data)*. IEEE. 2022, pp. 2610–2619.
- [34] Yiru Gong et al. “HLMD: Detecting Lateral Movement Using Heterogeneous Graph Model”. In: *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE. 2023, pp. 122–130.
- [35] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. “Datasets are not enough: Challenges in labeling network traffic”. In: *Computers & Security* 120 (2022), p. 102810. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2022.102810>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404822002048>.
- [36] Waqas Haider et al. “Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling”. In: *Journal of Network and Computer Applications* 87 (2017), pp. 185–192.
- [37] Xueyuan Han et al. “Unicorn: Runtime Provenance-Based Detector for Advanced Persistent Threats”. In: *Proceedings 2020 Network and Distributed System Security Symposium*. NDSS 2020. Internet Society, 2020. DOI: [10.14722/ndss.2020.24046](https://doi.org/10.14722/ndss.2020.24046). URL: <http://dx.doi.org/10.14722/ndss.2020.24046>.
- [38] Xueyuan Han et al. “Xanthus: Push-button Orchestration of Host Provenance Data Collection. CoRR abs/2005.04717 (2020)”. In: *arXiv preprint arXiv:2005.04717* (2020).
- [39] Hanan Hindy et al. “A taxonomy and survey of intrusion detection system design techniques, network threats and datasets”. In: (2018).
- [40] Hanan Hindy et al. “A taxonomy of network threats and the effect of current datasets on intrusion detection systems”. In: *IEEE Access* 8 (2020), pp. 104650–104675.

- [41] Grant Ho et al. “Hopper: Modeling and Detecting Lateral Movement”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3093–3110. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/ho>.
- [42] Microsoft Threat Intelligence. “Defending new vectors: Threat actors attempt SQL Server to cloud lateral movement”. In: (October 3, 2023). URL: <https://www.microsoft.com/en-us/securityblog/2023/10/03/defending-new-vectors-threat-actors-attempt-sql-server-to-cloud-lateral-movement/>.
- [43] Ryosuke Ishibashi et al. “Generating labeled training datasets towards unified network intrusion detection systems”. In: *IEEE Access* 10 (2022), pp. 53972–53986.
- [44] Alexander D. Kent. “Cybersecurity Data Sources for Dynamic Network Research”. In: *Dynamic Networks in Cybersecurity*. Imperial College Press, June 2015.
- [45] A. Kenyon, L. Deka, and D. Elizondo. “Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets”. In: *Computers & Security* 99 (2020), p. 102022. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2020.102022>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404820302959>.
- [46] Sean Michael Kerner. “CrowdStrike outage explained: What caused it and what’s next”. In: *TechTarget* (Oct. 2024). A CrowdStrike update caused a massive IT outage, crashing millions of Windows systems. Critical services and business operations were disrupted, revealing tech reliance risks.
- [47] Meejoung Kim and Inkyu Lee. “Human-guided auto-labeling for network traffic data: The GELM approach”. In: *Neural networks* 152 (2022), pp. 510–526.
- [48] Deepak Kushwaha et al. “Lateral Movement Detection Using User Behavioral Analysis”. In: *arXiv preprint arXiv:2208.13524* (2022).

- [49] Max Landauer et al. “A framework for automatic labeling of log datasets from model-driven testbeds for HIDS evaluation”. In: *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*. 2022, pp. 77–86.
- [50] Max Landauer et al. “Have it your way: Generating customized log datasets with a model-driven simulation testbed”. In: *IEEE Transactions on Reliability* 70.1 (2020), pp. 402–415.
- [51] Max Landauer et al. “Maintainable Log Datasets for Evaluation of Intrusion Detection Systems”. In: *IEEE Trans. Dependable Secur. Comput.* 20.4 (July 2023), pp. 3466–3482. ISSN: 1545-5971. DOI: 10.1109/TDSC.2022.3201582. URL: <https://doi.org/10.1109/TDSC.2022.3201582>.
- [52] Craig Laprade, Benjamin Bowman, and H. Howie Huang. “PicoDomain: A Compact High-Fidelity Cybersecurity Dataset”. In: *CoRR* abs/2008.09192 (2020). arXiv: 2008.09192. URL: <https://arxiv.org/abs/2008.09192>.
- [53] Fucheng Liu et al. “MLTracer: Malicious Logins Detection System via Graph Neural Network”. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, pp. 715–726. DOI: 10.1109/TrustCom50675.2020.00099.
- [54] Qingyun Liu et al. “Latte: Large-Scale Lateral Movement Detection”. In: *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. 2018, pp. 1–6. DOI: 10.1109/MILCOM.2018.8599748.
- [55] Xiaohan Ma, Chen Li, and Bibo Tu. “An Unsupervised Approach For Detecting Lateral Movement Logins Based On Knowledge Graph”. In: *2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. 2022, pp. 700–707. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom57177.2022.00095.
- [56] Anas Mabrouk et al. *Lateral Movement Dataset Generation (LMDG): Dataset and Documentation*. Available at: <https://github.com/AnasSalah9/LateralMovementDataset>. 2024. URL: <https://github.com/AnasSalah9/LateralMovementDataset>.

- [57] Mark Mazumder et al. “DataPerf: Benchmarks for Data-Centric AI Development”. In: *ArXiv abs/2207.10062* (2022). URL: <https://api.semanticscholar.org/CorpusID:250699092>.
- [58] Grant McDonald et al. “Ransomware: Analysing the impact on Windows active directory domain services”. In: *Sensors* 22.3 (2022), p. 953.
- [59] MITRE ATT&CK[®]. 2024. URL: <https://attack.mitre.org/>.
- [60] MITRE Corporation. *Caldera Documentation*. Accessed: 2024-10-31. 2024. URL: <https://caldera.readthedocs.io/en/latest/index.html>.
- [61] Basem Ibrahim Mokhtar et al. “Active Directory Attacks—Steps, Types, And Signatures”. In: *Electronics* 11.16 (2022), p. 2629.
- [62] Sowmya Myneni et al. “DAPT 2020-constructing a benchmark dataset for advanced persistent threats”. In: *Deployable Machine Learning for Security Defense: First International Workshop, MLHat 2020, San Diego, CA, USA, August 24, 2020, Proceedings 1*. Springer. 2020, pp. 138–163.
- [63] Sowmya Myneni et al. “Unraveled—A semi-synthetic dataset for Advanced Persistent Threats”. In: *Computer Networks* 227 (2023), p. 109688.
- [64] Kate O’Flaherty. “CrowdStrike Reveals What Happened, Why—And What’s Changed”. In: *Forbes* (2024). Senior Contributor; Kate O’Flaherty is a cybersecurity and privacy journalist. URL: <https://www.forbes.com/>.
- [65] Emilie Purvine, John R. Johnson, and Chaomei Lo. “A Graph-Based Impact Metric for Mitigating Lateral Movement Cyber Attacks”. In: *Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense*. SafeConfig ’16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 45–52. ISBN: 9781450345668. DOI: 10.1145/2994475.2994476. URL: <https://doi.org/10.1145/2994475.2994476>.
- [66] Mahdi Rabbani, Leila Rashidi, and Ali A Ghorbani. “A Graph Learning-Based Approach for Lateral Movement Detection”. In: *IEEE Transactions on Network and Service Management* (2024).

- [67] Markus Ring et al. “A survey of network-based intrusion detection data sets”. In: *Computers & security* 86 (2019), pp. 147–167.
- [68] Markus Ring et al. “Flow-based benchmark data sets for intrusion detection”. In: *Proceedings of the 16th European conference on cyber warfare and security. ACPI*. 2017, pp. 361–369.
- [69] Nithya Sambasivan et al. “”Everyone wants to do the model work, not the data work”: Data Cascades in High-Stakes AI”. In: 2021.
- [70] MohammadMoein Shafi et al. “Toward generating a new cloud-based Distributed Denial of Service (DDoS) dataset and cloud intrusion traffic characterization”. In: *Information* 15.4 (2024), p. 195.
- [71] Amit Sharma et al. “Advanced Persistent Threats (APT): Evolution, Anatomy, Attribution, and Countermeasures”. In: *Journal of Ambient Intelligence and Humanized Computing* 14 (May 2023), pp. 1–27. DOI: 10.1007/s12652-023-04603-y.
- [72] Hadi Shiravi, Ali Shiravi, and Ali A Ghorbani. “A survey of visualization systems for network security”. In: *IEEE Transactions on visualization and computer graphics* 18.8 (2011), pp. 1313–1329.
- [73] Florian Skopik et al. “Semi-synthetic data set generation for security software evaluation”. In: *2014 Twelfth Annual International Conference on Privacy, Security and Trust*. IEEE. 2014, pp. 156–163.
- [74] Christos Smiliotopoulos, Georgios Kambourakis, and Konstantia Barmpatsalou. “On the detection of lateral movement through supervised machine learning and an open-source tool to create turnkey datasets from Sysmon logs”. In: *International Journal of Information Security* 22 (July 2023), pp. 1–27. DOI: 10.1007/s10207-023-00725-8.
- [75] Robin Sommer and Vern Paxson. “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection”. In: *2010 IEEE Symposium on Security and Privacy*. 2010, pp. 305–316. DOI: 10.1109/SP.2010.25.

- [76] Branka Stojanović, Katharina Hofer-Schmitz, and Ulrike Kleb. “APT datasets and attack modeling for automated detection methods: A review”. In: *Computers & Security* 92 (2020), p. 101734. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2020.101734>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404820300213>.
- [77] Xiaoqing Sun and Jiahai Yang. “HetGLM: Lateral Movement Detection by Discovering Anomalous Links with Heterogeneous Graph Neural Network”. In: *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. 2022, pp. 404–411. DOI: 10.1109/IPCCC55026.2022.9894347.
- [78] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. “Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Methods”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40.5 (2010), pp. 516–524. DOI: 10.1109/TSMCC.2010.2048428.
- [79] Ngan Tran et al. “Data Curation and Quality Evaluation for Machine Learning-Based Cyber Intrusion Detection”. In: *IEEE Access* 10 (2022), pp. 121900–121923. DOI: 10.1109/ACCESS.2022.3211313.
- [80] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. “Unified Host and Network Data Set”. In: *Data Science for Cyber-Security*. World Scientific, Nov. 2018. Chap. Chapter 1, pp. 1–22. DOI: 10.1142/9781786345646_001. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9781786345646_001. URL: https://www.worldscientific.com/doi/abs/10.1142/9781786345646_001.
- [81] Martin Ussath et al. “Advanced persistent threats: Behind the scenes”. In: *2016 Annual Conference on Information Science and Systems (CISS)*. 2016, pp. 181–186. DOI: 10.1109/CISS.2016.7460498.

CHAPTER 4

Conclusion

In this work, we have addressed critical gaps in the existing datasets related to lateral movement (LM) attacks within the context of advanced persistent threats (APTs). Our analysis of current cybersecurity benchmark datasets revealed significant challenges, including the scarcity of LM instances, limited diversity in LM techniques, and the inadequacy of attack paths, particularly those involving multiple hops. Additionally, many existing datasets fail to capture the complexities of cloud-based lateral movement scenarios or present outdated attack patterns, thus limiting their usefulness for training robust detection models. We identified the necessity for comprehensive datasets that incorporate a wider range of lateral movement activities and more varied attack vectors to improve detection accuracy and model generalization.

To address these challenges, we developed a new benchmark dataset focused exclusively on lateral movement attacks. This dataset was designed to overcome many of the limitations of existing datasets, offering a broader range of attack types, a more varied time frame, and a comprehensive set of data sources. Our qualitative analysis of this dataset confirms its value for training and evaluating LM detection models, as it captures the complexity and diversity of attack behaviors seen in real-world scenarios.

A key contribution of this work is the Lateral Movement Dataset Generator (LMDG) framework, which provides a reproducible toolset for generating high-quality LM and APT datasets. The LMDG framework automates the generation of benign data, the execution of attack scenarios, and, crucially, the labeling of attack-related

events in system and network logs. One of the most significant challenges in LM dataset creation is the accurate labeling of benign hosts involved in malicious actions. To address this, we introduced the novel "process tree labeling" technique, which significantly improves the accuracy and precision of automatic labeling in comparison to existing methods.

While our framework represents a significant step forward, it is not without its limitations. The complexity of testbed construction, the challenges of automating attack execution, and the dependency of our labeling approach on client-server architectures present barriers to scalability and broader applicability. These limitations are acknowledged and discussed in the context of future work, which will focus on extending the framework to support more diverse environments and incorporating more rigorous quantitative evaluation metrics to ensure the dataset's reliability.

Ultimately, this work contributes to the advancement of lateral movement detection by providing a more robust and realistic dataset for research and model training. It also opens the door to developing more comprehensive datasets that encompass the full lifecycle of APT attacks. Such datasets will be invaluable for the cybersecurity community, supporting the development of more effective, holistic threat detection systems capable of defending against increasingly sophisticated attack strategies.

VITA AUCTORIS

NAME: Anas Mabrouk

PLACE OF BIRTH: Giza, Egypt

YEAR OF BIRTH: 1999

EDUCATION:

Helwan University, B.Sc in Computer Science, Egypt, 2022

University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2024