Article

Revisiting the Detection of Lateral Movement through Sysmon

Christos Smiliotopoulos[1], Konstantia Barbatsalou[1], Georgios Kambourakis[2,*]

[1] Department of Information and Communication Systems Engineering, University of the Aegean, Karlovasi 83200, Samos, Greece; csmiliotopoulos@aegean.gr (C.S.); tbarbatsalou@gmail.com (K.B.)
[2] European Commission, Joint Research Centre, Ispra 21027, Italy; georgios.kampourakis@ec.europa.eu (G.K.)
* Correspondence: georgios.kampourakis@ec.europa.eu; gkamb@aegean.gr

Abstract: This work attempts to answer in a clear way the following key questions regarding the optimal initialization of the Sysmon tool, towards the identification of Lateral Movement in the MS Windows ecosystem. First, from an expert's standpoint and with reference to the relevant literature, what are the criteria of determining the possibly optimal initialization features of the Sysmon's event monitoring tool, which are also applicable as custom rules within the config.xml configuration file? Second, based on the identified features, how can a functional configuration file, able to identify as many LM variants as possible, be generated? To answer these questions, we relied on the MITRE ATT&CK knowledge base of adversary tactics and techniques, and focused on the execution of the nine commonest LM methods. The conducted experiments, performed on a properly configured testbed, suggested a great number of interrelated networking features, that were implemented as custom rules in the Sysmon's config.xml file. Moreover, by capitalizing on the rich corpus of the 870K Sysmon logs collected, we create and evaluate in terms of TP and FP rates an extensible Python .evtx file analyzer, dubbed *PeX*, which can be used towards automatizing the parsing and scrutiny of such voluminous files. Both the .evtx logs dataset and the developed PeX tool are provided publicly for further propelling future research in this interesting and rapidly evolving field.

Keywords: Lateral Movement; Sysmon; Dataset; Attacks; Network Security; Hacking

More and more, the community is witnessing cyberattacks during which malware infects a machine and it quickly diffuses laterally within the network infrastructure of an organization. Its propagation, whether undiscovered or untreated can provoke catastrophic events for the systems as a whole. For effectively dealing with such incidents in a prompt manner, a targeted and fresh approach is a necessity. Once the threat is remediated and the related to the attack's log evidence are collected, the impact calculation of the damage will be used as the basis for the constitution of remedial measures towards an effective Endpoint Detect and Response (EDR) policy.

While the configuration of a network targeted by an adversary varies depending on its structure, there are some common patterns regarding Lateral Movement (LM) methods. At first, typically, the attacker concentrates on the identification and enumeration of the targeted system, in parallel with the infiltration of crucial information with tools such as Mimikatz, ipconfig, systeminfo, and others. The initial target is chosen with meditation, as this will be the first step for the rest of the LM to be expanded. Credential dumping follows; this will permit the perpetrator to acquire the necessary credential information, leveraging tools like Mimikatz, pwdump, LazagneProject, or even malware in an effort to infect the targeted host and acquire administrative access. Common techniques presented in MITRE's records [1] reveal that in such conventional similar assault cases, the adversaries repeatedly leverage a limited number of penetration tools. Therefore, the key points that this paper aims to address are narrowed to the fundamental and technical particles of each attack, specifically related to the co-called "5Ws" analysis, namely Who - What - When - Where and Why.

Precisely, the work at hand relies on the execution of the most frequently encountered LM techniques based on their impact, as presented on MITRE's ATT&CK database of adversaries tactics and techniques [1]. Particularly, we exploit a properly designed testbed via nine diverse LM techniques, namely "Exploitation of Remote Services (ERS)" (four variants of ERS attack), “Pass the Hash (PtH)”, “Pass the Ticket (PtT)”, “Golden Ticket (GT)”, “Silver Ticket (ST)” and “Post Exploitation on Stored Passwords with (LZP)”. Our aim is to provide concrete answers to the following key questions: (a) Based on related theory and domain expertise’s best practices, are there any solid criteria for determining the most effective, upon LM identification, rule-based features of the System Monitoring tool (Sysmon)?, and (b) how the proposed criteria combined with the extracted features could lead to the formation of a top-notch rule-based and event-driven methodology, which generalizes on the identification of any LM attack?. Overall, the main contributions of this work vis-à-vis the relevant literature are summarized as follows:

- From a defender’s viewpoint, we elaborate on the outcomes of the executed experiments in an attempt to define solid criteria, which act as signatures per examined attack.
- Based on the derived criteria, the most impactful features were extracted as related to Sysmon’s identification of an adversary moving laterally.
- The proposed criteria along with their related features are evaluated based on their effectiveness on potential lateral incident’s identification through Sysmon.
- A rule-based methodology is proposed that is dedicated to the identification, verification, and categorization of each lateral technique.
- The proposed methodology is implemented in a Python analyzing tool, dubbed “PeX”. The tool along with the Sysmon logs large dataset collected during the experiments are made publicly available to the community [2].

The remainder of this paper is structured as follows. Section 2 discusses the related work, whereas Section 3 encloses a general description of pertinent to this work fundamental LM techniques. Section 4 presents the testbed, while Section 5 details the commonest, based on their final impact, LM attack techniques. The proposed data-driven methodology is provided in Section 6. Section 7 focuses on the presentation of the *“PeX”* tool. The last section concludes and suggests directions for future work.

2. Related Work

The current section provides a brief review of the key pertinent literature on the subject. The concentration is on the methodology of each relevant work regarding event-driven identification of the most impactful LM techniques through Sysmon. Particularly, we focus on the logs collection of the specified examined attacks, the utilized rule-based policy, and the impact upon the successful incidence response. To facilitate the parsing of the relevant literature, Table 1 recaps the relevant characteristics of every work included in this section.

JPCERT/CC, the first Computer Security Incident Response Team (CSIRT) established in Japan, was also one of the first organisations to conduct a full-fledged survey regarding the categories of logs produced during the execution of LM [3]. The commonest at that time LM techniques were executed against both compromised servers and targeted clients, related to each attack method. Logging information was collected and categorised via Sysmon [4] and MS Windows audit-policy. The work concluded with the proposition of a log-oriented MS Windows regulatory policy, allowing the optimal collection of useful information related to potential LM methods. The final report was published in June 2017 and updated on v2 [5] in Dec. of the same year. On the downside, considering the exponential occurrence of LM events and the daily lessons learned from the Russo-Ukrainian war, this survey is considered quite outdated to cope with MITRE’s [1] database of malicious LM tactics and techniques on real-life observations. For instance, the aforementioned work can be extended to incorporate LM events in cloud and satellite communication facilities like the incident of Feb. 2022 against Viasat’s KA-SAT network [6], resulting in a targeted interruption of the satellite broadband services across the Ukraine territory and other European countries.

Mavroeidis et al. [7] introduced a data-driven threat classification methodology, which relies on the continuous analysis and assessment of aggregated Sysmon logs. The first part of this work was dedicated to the development of their proposed Cyber Threat Intelligence Ontology (CTIO), which was based on the Cyber Threat Intelligence (CTI) model also published by them in [8]. The second part of that paper proposed a threat assessment system that incorporated CTIO towards the classification of event-logs generated by Sysmon in four distinct threat categories, namely high, medium, low, and unknown. Despite the promising results, the proposed threat assessment methodology is prone to long delays when it comes to voluminous logging data on real-life scenarios.

Berady et al. [9] presented a threat hunting model based on the common acceptance of the conclusion that the attacker and defender should mutually understand each other. The analysis was performed from both an offensive and defensive perspective. Offensive experiments were conducted upon the *APT29* dataset, that is included in the Mordor Project of pre-recorded event-related logs of malicious activities. The two-staged attack emulation scenario incorporated initially the exfiltration of sensitive data for the targeted host, and then, the compromise of the target via an injected toolkit. The network activity of the targeted host was monitored with Sysmon to produce a dataset comprising two-days log's recording. The finally proposed "Indicators of Compromise" enhanced the defender's common knowledge of offender's best practices, improving seemingly the preventive identification of threats. Nevertheless, it seems that the proposed model is incapable of dealing effectively with the vast existence of false-positives due to the lack of Sysmon's rule-based configuration policy.

The work in [10] proposed a Dynamic Link Library (DLL)-oriented method for malicious files detection towards logs collected by Sysmon. Numerous DLL files were collected and analyzed from four different tools namely, China Chopper, Mimikatz, PowerShell Empire, and HUC Packet Transmitter, to reveal the existence of significant differences of the specific files among various versions of the MS Windows Operating System (OS) and the aforementioned tools. The work concludes with the extraction of a list with the commonly loaded DLLs per tool, regardless the Windows OS environment and the introduction of a logging detection specific method based on the open-source ELK Stack. The researchers in [11,12] also attempted to introduce a DLL-oriented malware detection methodology through the analysis of potentially malicious files on sandbox environments, like *Cuckoo*. However, only the authors in [10] proposed a list with the most commonly appeared DDL files related with malicious intrusion tools. Additionally, the contributions in [11,12] relied on classification techniques with Machine Learning (ML) algorithms, which at this point fall out of the scope of the current research. It can be said that the enhancement of the DLL detection policy with particularly focused on each LM method Sysmon event logs, will expand the detection accuracy and diminish the false positive results.

The ELK Stack was also used by the authors in [13,14] for the analysis of massive log records and the identification of malicious behaviour. Specifically, the work in [13] implemented a *logstash* massive data processing pipeline to collect a critical mass of logs, whereas [14] generated Sysmon logging events. In both cases, the ELK stack was utilised for log file's iteration and identification of malicious patterns. On the downside, both works were tightly bonded to ELK stack practices, neglecting the need of a general purpose rule-based EDR system.

The work in [15] focused on Advance Persistent Threat (APT) detection of the Mimikatz password stealing tool, while utilized in the context of LM. To increase the true positive rates, the authors implemented *Mutex* memory objects in parallel with the identification of Mimikatz related DLL files. It can be argued that even with the introduction of *Mutex-oriented* DLL files analysis, Mimikatz tool can be deliberately obfuscated by the adversary, thus evading identification by even advanced EDR and IDS systems.

Last but not least, the work in [16] considered the detection of malware Application Programming Interface (API) call patterns towards the identification of anomaly behavioural signatures. Despite the high true positive rates, the proposed methodology affects the performance of the examined system and omits to incorporate Sysmon event logs. On the other hand, the authors in [17] suggested a LM detection system, dubbed "Hopper", that is fed by real-life generated logs. User's login activity was tracked and outlined through a graph of interrelated logins among the implicated hosts. This contributes in anomalies detection among logins, referring to LM attacks. On the flip side, despite "Hopper's" effectiveness, the system ignores logs generated with Sysmon. In this respect, the aforementioned two papers are considered out of the scope of this study.

Table 1: Summary of the most important aspects of the works included in Section 2. The works are presented in chronological ascending order.

RELATED WORK

Title	**Year**	**Summary**
A Novel Approach to Detect Malware Based on API Call Sequence Analysis [16]	2015	Detection of malware API call patterns towards the identification of anomaly behavioural signatures
Detecting lateral movement through tracking event logs (v1 & 2) [3,5]	2017	Execution of LM through well-known penetration testing tools. Collection of logs with Sysmon and MS Windows Audit Policy. Categorization of logs per attack and proposition of optimal MS Windows infiltration settings for effective identification of LM activity.
Data-driven threat hunting using Sysmon [7]	2018	Data-driven threat classification methodology based on aggregated logs created by Sysmon. Proposed a threat analysis system which is based on the developed by the authors CTI ontology.
Lateral movement detection using ELK stack [14]	2018	A way to generate event-logs with Sysmon, related to the execution of various LM attacks and the associated malicious tools on MS Windows-based environments. Implementation of ELK stack towards the analysis of possible abnormalities in the collected logs due to existence of LM.

Continued on next page

Table 1 – continued from previous page

RELATED WORK		
Title	**Year**	**Summary**
Real-time detection system against malicious tools by monitoring DLL on client computers [10]	2019	Proposed a DLL-oriented method for malicious files detection through logs collected by Sysmon [4]. Common DLL list per malicious tool, independent to MS Windows OS version.
Detecting Mimikatz in lateral movements using Mutex [15]	2020	APT detection of Mimikatz, while utilized in LM. Implementation of Mutex memory objects in conjunction with DLL files analysis. Mimikatz's misidentification while deliberately obfuscated.
From TTP to IoC: Advanced persistent graphs for threat hunting [9]	2021	Threat hunting model which evaluates Sysmon's logs from both an offender's and defender's perspective. Based on indicators of compromise, proactive threat detection is possibly enhanced. A high rate of false positives due to the absence of rule-based Sysmon's configuration.
Hopper: Modeling and detecting lateral movement [17]	2021	System for LM attack detection based on real-life generated logs. Login activity is tracked and outlined through a graph of interrelated logins among the implicated hosts to conclude in the detection of anomalies among logins referring to LM.
Network forensics investigation in virtual data centers using ELK [13]	2021	Generating log records with *Logstash*. Network forensic analysis via the implementation of Elastic Search and towards the identification of RDP LM-related attacks, Ransomware, Data Exfiltration, etc. as part of a criminal investigation.

3. Preliminaries on Lateral Movement

This section comprises a descriptive presentation of the most prominent and fundamental techniques with which an adversary may move laterally and compromise computing systems and accounts in Small Office Home Office (SOHO) or corporate networking environments. LM consists of all the attacking practices leveraged by perpetrators to compromise and control remote computing systems in basic and complex networking infrastructures. For an attacker to compromise a targeted host, initial access is gained to any network component no matter its rank of credential rights towards the targeted system. This is achieved through a reconnaissance stage, involving the continuous and iterative enumeration of the systemic parts of the network to identify accounts with potentially elevated user privileges.

LM techniques can be executed through the combination of a large repertoire of offensive tools. Regarding user's account credential exploitation, the adversary may achieve to leverage the authentication mechanisms of the targeted network via Metasploit framework or even malware propagation. Additionally, advanced hashing compromising techniques, such as PtH and GT, may jeopardize confidentiality and integrity of the transmitted and received information on the targeted host.In brief, during the execution of LM, the aggressors occupy assets account's credentials of low or medium access level of the targeted network, escalating their privileges up to their final target. This is done through various techniques, which are detailed in Sections 5 and 6.

As depicted in Figure 1, the boundary between the internal network's perimeter and the outside world is presented as a horizontal line. The upper half of the line comprises the outside world, whereas the environment under the line represents the closed, say, corporate network. For a malevolent intruder to penetrate a network, the movement should be executed vertically and towards the depicted horizontal boundary line. Mostly, this kind of movement is called "North to South" [18]. When credentials are exposed, an anchor point is created, and the intruder can maneuver with privilege access horizontally and around the network components for gradually reaching and compromising the targets ("East to West" movement). It should be noted that, while it seems normal for a network component to communicate with certain terminals inside the same network domain, it is suspicious for the node to continuously attempt various connections to open ports, or even try to take advantage of credential acquisition services, or attempt an account connection with a username or password for the first time. On the other hand, from a defender's viewpoint, LM usually appears as an abnormality of network activity.

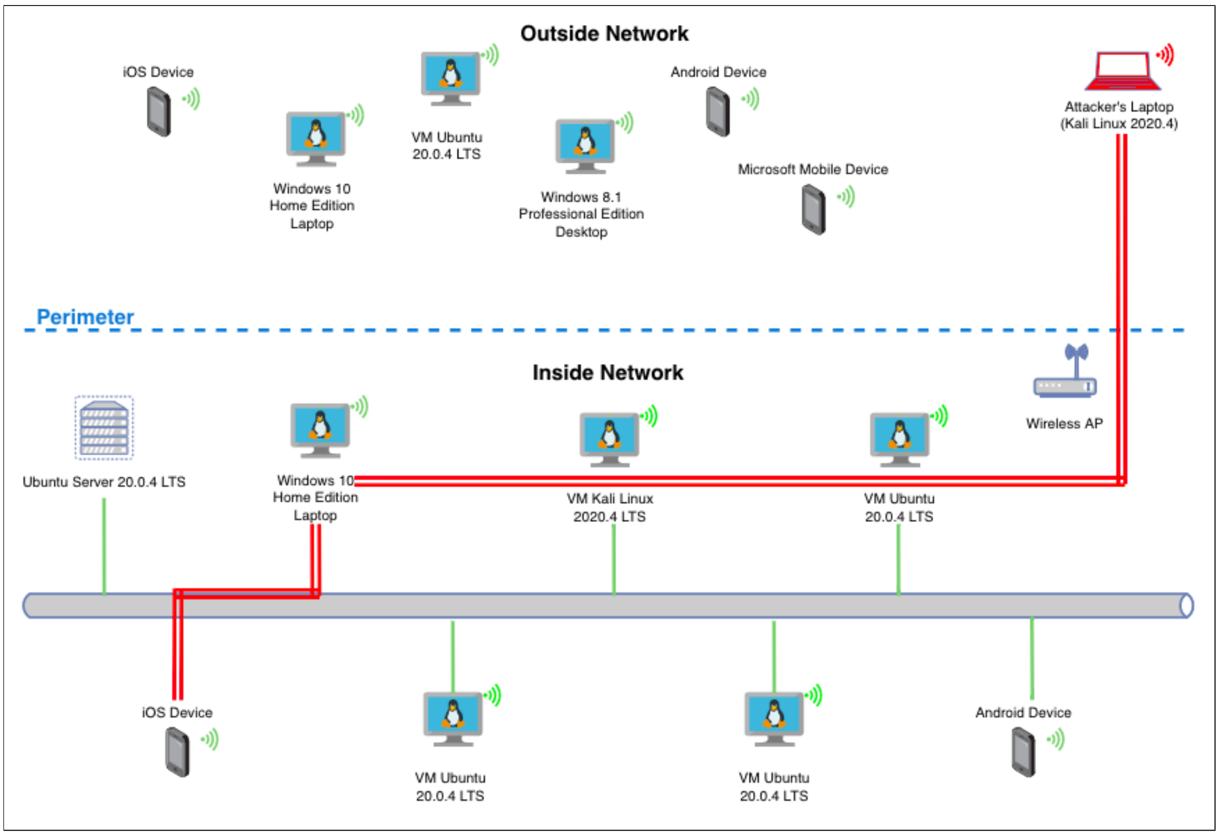


Figure 1. A bird's-eye view of a typical LM.

Overall, there are two fundamental methodologies for moving laterally within a targeted network:

- The first method refers to network mapping and identification of the special characteristics of each distinct component along with potential vulnerabilities. During the reconnaissance phase, many OS's built-in tools can be used. For instance, the ipconfig and ifconfig commands reveal IP configurations and localization information, while Netstat summarizes the most current network connections along with their users. Additionally, the local routing table tracks all the interconnected network paths, while the Address Resolution Protocol (ARP) cache reveals the matching of network IP addresses to their equivalent link layer (MAC) addresses. Vulnerability assessment and mapping tools are also in the attackers' quiver, namely Nmap or its commercial equivalent Nessus [19]. Once the relevant information is gathered, the adversary is able to intrude and move laterally.
- The second customary method refers to the use of phishing techniques towards the acquisition of account credentials by any means. To this end, the opponent can take advantage of tools, say, Metasploit Pro, to craft numerous phishing emails that have been perfectly cloned to appear as originating from a legitimate source. Additionally, keylogger tools, including Revealer Keylogger Free, KidLogger, and BlackBox Express, provide a powerful equivalent solution for activity monitoring and screen recording at the victim's side. Mimikatz, presented in Subsection 6.2, is another powerful credential exploitation and hash dumping penetration testing tool.

4. Testbed

For the purposes of log file gathering (in .csv, .evtx, .xml form), as depicted in Figure 2, we created a mixed testbed comprised of both virtual and physical stations that realistically emulates a typical SOHO environment. In total, eight different physical and Virtual Machines (VMs) were utilized. Three of them were used as client stations (1 Ubuntu Desktop 20.0.4 LTS, 1 Kali Linux Desktop 2020.4 LTS, 1 MS Windows 10), one both as client station and host PC on which VMs have been installed, and two smartphones as clients (1 Android smartphone Samsung A8 and 1 iOS iPhone XR). Moreover, one physical laptop (MacBook Air) simulated the attacker's interface. Finally, one MS Windows Server 2019 64-Bit was created as a VM station to dissemble the malicious LM's final target. The wireless Access Point (AP) was a SERCOM Speedport Plus router. The SOHO's network topology is presented in detail in Figure 2, while the role and the characteristics per machine are included in Table 2.

To simulate as realistically as possible the workload of a typical SOHO network, the *SYSMON_SET domain* was created within a MS Windows 2019 Server VM. Six client accounts were configured and initialized via the *MS Windows Server Account Management* service and joined the SYSMON_SET domain. For the needs of the execution of the various credential exploitation attacks detailed in Section 6.2 each of the server's accounts was granted domain administrative privileges in advance.

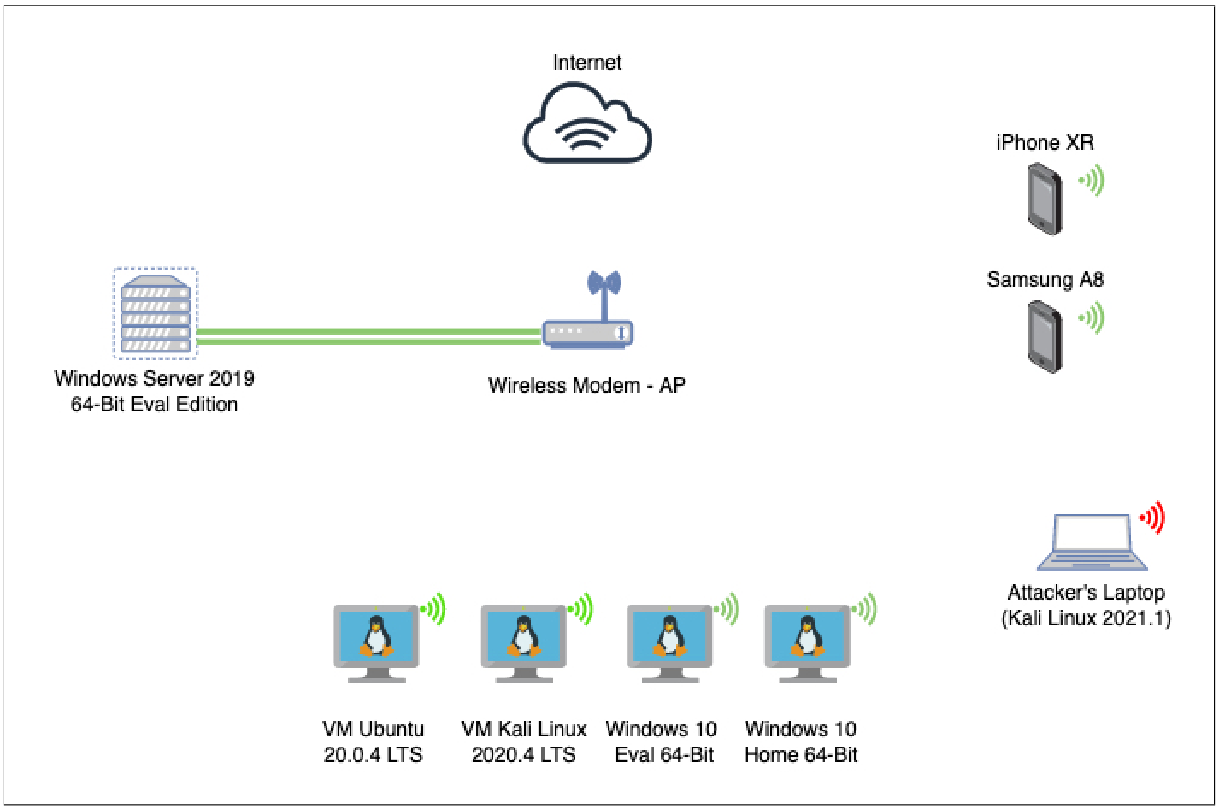


Figure 2. High-level view of the testbed network topology. The MS Windows Server 2019 uses a wired connection with the AP, presented as a double green line. The network traffic flow for disparate categories of VM and physical stations is shown in different colour, namely green for WiFi traffic and red for the attacker

5. Lateral Movement Categories on Windows Environments

As already mentioned, LM attack techniques are leveraged by cyber criminals to intrude and remotely control network systems and hosts. Network reconnaissance and target identification precedes to be followed by the penetration of the protected networking perimeter. For the malicious access to be successful and the stealthiest possible, the lateral rotation around various network nodes is a necessity. It is achieved with the installation of remote accessing tools on systems, the credentials of which were successfully extracted, and then it escalates incrementally to other hosts. Based on the revised MITRE ATT&CK open-access database [1], among the many available techniques and practices, in the context of this paper, the nine commonest were chosen based on their impact. These attacks were applied on the testbed's nodes of section 4, and are described in the following subsections.

5.1. Exploitation of Remote Services

Malicious users may exploit MS Windows systems' remote services to gain unauthorized access to the various nodes of a network. This kind of exploitation is successful when the adversary takes advantage of a revealed vulnerability on the targeted system. This might be a false implementation in the source code of an application or Windows service, or even a problematic initialization of the Windows OS kernel that will allow the execution of malicious code and the approval of unauthorized remote network access. To determine whether the system is in vulnerable state, the assailant makes use of numerous information gathering techniques to reveal unpatched OS components or even the lack of updated IDS and antivirus software. Among the most common, port scanning and vulnerability assessment tools are included, which are loaded and installed remotely on the system under attack. Server nodes are considered the most high-value targets

Table 2: Summary of key technical characteristics related to the machines utilized during the execution of the LM experiments.

TECHNICAL CHARACTERISTICS AND UTILIZATION OF EACH STA (PHYSICAL AND VM)										
Station	**Type**	**Brand**	**Operating System**	**Kernel_Version**	**WNIC**	**Driver_Version**	**CPU/RAM**	**IP_Address**	**MAC_Address**	**Username**
AP	Wireless Router	SERCOM Speedport Plus	Speedport Plus	09022001.00.031 OTE2	SERCOM Speedport Plus	N/A	N/A	"80.107.57.231 192.168.1.1"	7C:8F:DE:46:66:11	N/A
Client STA 1	VM	Custom	Ubuntu Linux 20.0.4 LTS (Focal Fossa)	Linux 5.8.0-53-generic x64	Intel(R) Wireless-AC 9560 160MHz	22.0.1.1	Intel® Core™ i7-9750H CPU @ 2.60 GHz 2.59 GHz (x2 VM Processors), 2 GB DDR4 (VM Memory Allocated) RAM	192.168.1.6	08:00:27:4e:4e:e6	adminchr
Client STA 2	VM	Custom	Kali Linux 2021.1 LTS	Linux 5.10.0-kali7-amd64	Intel(R) Wireless-AC 9560 160MHz	22.0.1.1	Intel® Core™ i7-9750H CPU @ 2.60 GHz 2.59 GHz (x2 VM Processors), 2 GB DDR4 (VM Memory Allocated) RAM	192.168.1.11	08:00:27:3e:b5:a7	nicksaitas
Client STA 3	VM	Custom	Windows 10 Evaluation Edition x64-Bit	2004	Intel(R) Wireless-AC 9560 160MHz	22.0.1.1	Intel® Core™ i7-9750H CPU @ 2.60 GHz 2.59 GHz, 16 GB DDR4 RAM	192.168.1.7	08:00:27:3d:c5:a9	chr.smilio
Client STA 4	Laptop	HP Omen 17” Intel i7	Windows 10 Home x64	2004	Intel(R) Wireless-AC 9560 160MHz	22.0.1.1	Intel® Core™ i7-9750H CPU @ 2.60 GHz 2.59 GHz, 16 GB DDR4 RAM	192.168.1.28	98:AF:65:32:E8:21	laptop-ropr18ak_chrsm
Windows 2019 Server	VM	Custom	Windows 2019 Server x64-Bit Evaluation Edition	1809	Intel(R) Wireless-AC 9560 160MHz	22.0.1.1	Intel® Core™ i7-9750H CPU @ 2.60 GHz 2.59 GHz (x2 VM Processors), 2 GB DDR4 (VM Memory Allocated) RAM	192.168.1.8	08:00:27:4f:d3:32	christossmiliotopoulos
Client STA 5	Smartphone	iPhone XR	iOS	14.4.2	Wi-Fi 802.11 a/b/g/n/ac, dual-band, hotspot	3.02.02	Hexa-core (2x2.5 GHz Vortex + 4x1.6 GHz Tempest), 64GB RAM	192.168.1.3	06:16:EC:81:F9:68	iPhone-Christos
Client STA 6	Smartphone	Samsung A8	Android 9	Knoxx 3.3	Wi-Fi 802.11 a/b/g/n/ac, dual-band, hotspot	SM-A530F	Exynos 7 Octa 7885	192.168.1.13	3C:DC:BC:7F:73:B7	A8-Maraki

during the execution of LM. Mostly, they comprise the final destination of an attacker after gaining initial access to a network node and hopping from one system to another.

Among the most used network scanning procedures, based on MITRE's ATT&CK list [1], CrackMapExec (CME) is a Python post-exploitation network penetration testing tool. It is executed repetitively through PowerShell and enumerates Windows Active Directory's information, specifically those relevant to the execution of LM. In more detail, it allows the execution of custom Python scripts to brute-force and dump credentials. Another method provided is password-spraying with single pre-guessed username and password lists and well-known dictionaries as an attempt to compromise a valid user account on the network under attack. Moreover, several other open-source tools presented in [1] can be used in conjunction with LM methods to reveal EternalBlue, namely DoublePulsar and SMBTouch. MITRE's ATT&CK list continues updating with references of the most identified and dangerous network scanning and credential compromising techniques for revealing open port and malware vulnerabilities.

There are many well-known vulnerabilities concerning the MS Windows OS various versions. Server Message Block (SMB) Authenticated Remote Code Execution (RCE) and Remote Desktop Protocol (RDP) are two of the most popular misconfigurations. Within the context of this paper, exploitation of remote services was performed against the testbed described in Section 4. Precisely, the EternalBlue vulnerability was used in three different ways to exploit the SMB of an unpatched version of Windows Server 2019, which was left vulnerable on purpose, while RDP exploit was tested with BlueKeep (CVE-2019-0708) RCE. Overall, four different instances of the Exploitation of Remote Services attack were executed. First, SMB was exploited via the Metasploit's ms17-010 module. Next, the SMB EternalBlue vulnerability was used towards the final execution of the remote arbitrary commands on the targeted system via Metasploit's *smb_login* auxiliary service. To expand the analysis of the remote exploitation on the targeted server, the testbed was infected with the worm-like *WannaCry* malware. To allow the propagation of the malware, User Account Control (UAC) of the server was disabled remotely with the use of the *smbexec 2.0* tool. Finally, Metasploit's CVE-2019-0708 dedicated module was utilized to leverage the targeted Server's RDP misconfiguration.

5.2. Pass the Hash Credential Override

Hash transmission is a method that provides account identification and credentials' integrity check without the requirement for any username or password in plaintext form. This method overrides traditional user's authentication, which requires a personal access password in plaintext form, and proceeds directly to host's approval with the use of hash algorithm encrypted character chains. Hash distribution instead of plaintext account credentials was considered a tough-to-crack identification and authentication method. Despite the robustness of the aforesaid authentication mechanism, penetration techniques have been developed to allow the collection and extraction of all the related to valid accounts hashes. These techniques fall under the general category of "Credential Access". MITRE ATT&CK [1] describes this attack vector as the *keylogging* and *credential dumping* techniques, making LM more effective and harder to be identified in time.

PtH is a method for achieving successful authentication without the possession of the original credentials (usernames and passwords) in cleartext form. It is considered a very successful and targeted attack for bypassing traditional authentication procedures via stolen hashing credentials. Through PtH, access control services of Windows can be compromised, considering an adversary's LM as legit. PtH is a credential theft technique and LM attack at the same time, with which the attacker can leverage the challenge-and-response nature of the Windows New Technology LAN Manager (NTLM) authentication procedures, utilized from the network's hosts in place of traditional plaintext credentials. With standard security protocols, NTLM hashes change only when the plaintext passwords are changed. This is the feature that makes PtH ideal for threat actors to move laterally across a network.

Besides that, Local Security Authority Subsystem Service (LSASS) process, is a fundamental part of Windows credential administration system which is also related to PtH. The aforementioned process regulates the security policy that rules Windows OS verification and validation procedures. If the *lsass.exe* process fails to verify a user, the credentials for the accounts related to them are revoked automatically. For easy reference, the Windows Kerberos hashing authentication procedure along with the NTLM hash extraction vulnerability are illustrated in Figure 3.

A basic NTLM hash is comprised by four distinct part each separated by a semicolon (:), presented as follows:

```
admin2:1000:aad3b435b51404eeaad3b435b51404ee:7178d3046e7ccfac0469f95588b6bdf7:::
```

- The first part (admin2:) is the account's username.
- The second part (:1000:) is the numerical identifier of the relative account to the username from the first part.
- The next part (:aad3b435b51404eeaad3b435b51404ee:) is the LM hash. This type of hash was used with Windows OS prior to v10, and is kept for reasons of reverse compatibility with legacy systems that are still in use.

- The final part (:7178d3046e7ccfac0469f95588b6bdf7:::) of a hash is the NTLM hash. This is the redesigned version of hashes used in Windows LSASS. This is the feature that makes NTLM hashes robust against common brute force credential compromising techniques.

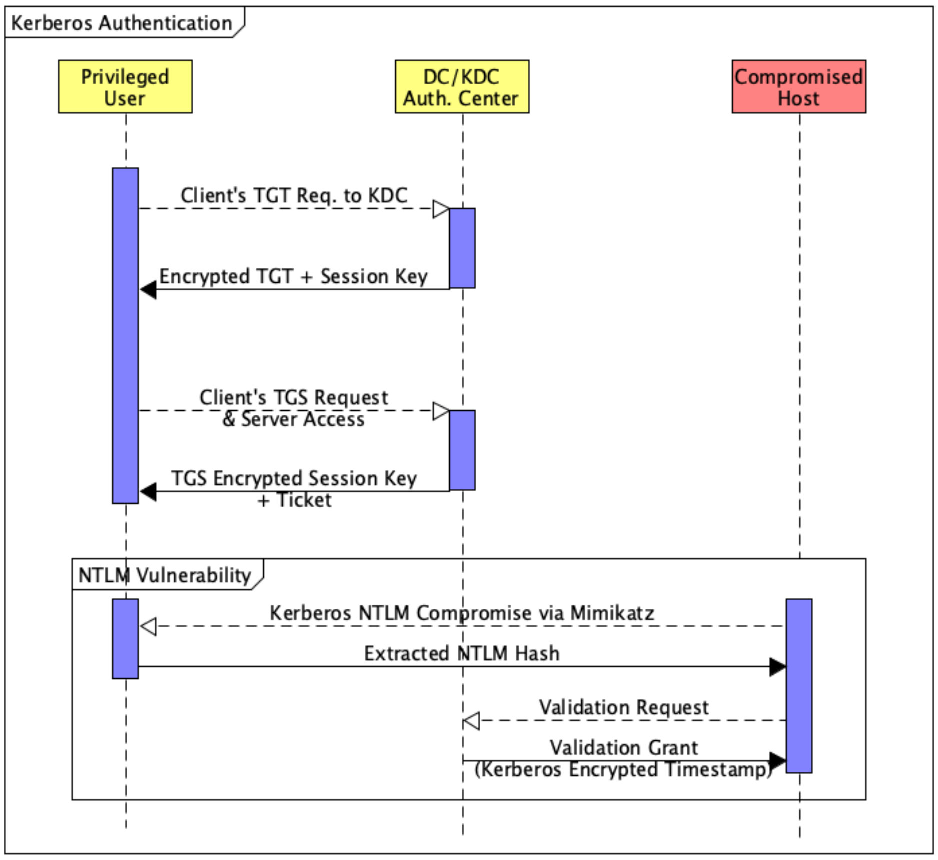

Kerberos Authentication
Privileged User
DC/KDC Auth. Center
Compromised Host
Client's TGT Req. to KDC
Encrypted TGT + Session Key
Client's TGS Request & Server Access
TGS Encrypted Session Key + Ticket
NTLM Vulnerability
Kerberos NTLM Compromise via Mimikatz
Extracted NTLM Hash
Validation Request
Validation Grant (Kerberos Encrypted Timestamp)

Figure 3. High-level view of Kerberos Authentication procedure, along with the NTLM hash extraction vulnerability, that makes KDC prone to the Hashing Exploitation Attacks of Section 6.2, namely, PtH, PtT, Golden/Silver Ticket.

5.3. Pass the Ticket Credential Override

By exploiting the Pass-the-Ticket (PtT) method an adversary may move laterally within a network by using stolen Windows Kerberos Tickets as legitimate credentials. PtT is similar to PtH and can be used for the authentication of an external malicious user during their first LM steps towards a remote system. Since 2016, Windows Server OS versions incorporate Kerberos v5, along with all the published extensions related to public key authentication and data authorization. A Kerberos authentication client is accessed via the windows Security Support Provider Interface (SSPI). Windows Active Directory Domain Services is the major requirement for the Kerberos Key Distribution Center (KDC) implementation within a domain, whose the user's authentication procedure is structured upon Winlogon Single Sign-On (SSO) architecture. PtT is based on Windows OS's Local LSASS procedural architecture, which stores and handles Kerberos tickets. It is related to the extraction of the Kerberos Service Tickets and Ticket Granting Tickets (TGT), depending on the level of access of the compromised host. Recall that a service ticket permits the access to a particular process, whereas a TGT is responsible to request the aforesaid service tickets from Kerberos Ticket Granting Service (TGS) to acquire access to any resource that the targeted host has access to. Specifically, for this attack to be successful the targeted host needs to communicate with LSASS and retrieve the legitimate tickets on its possession. If the attacked host is related with a non-administrative user, then only one ticket will be retrieved, whereas if administrative accounts are compromised, all the tickets related to the targeted server can be harvested.

The tools mostly related to this attack are Mimikatz and Rubeus. The former has already been mentioned in subsection 5.2. The latter is a C# hacking tool similar to Mimikatz that was created for raw Kerberos interactions and tickets' malicious exploitation. Within the context of this survey, Mimikatz was the preferred tool for the execution of the PtT attack.

5.4. Golden Ticket

Golden Ticket is another Active Directory attack related to Kerberos authentication exploitation techniques. While access with administrative privileges is important for the adversary, persistence during lateral movement within a targeted network is the real challenge. This aspect is crucial for an opponent to reach its target within network environments in which the domain's admin password changes frequently.

Golden ticket migrates features from both PtH and PtT LM methods. Precisely, it exploits Kerberos authentication protocol lack of validation procedures during the impersonation of a legitimate host. This is due to the design of the Kerberos protocol to allow users that hold a TGT for a session to be considered as trusted for any other network resource related to ticket authentication.

In the context of this work, Mimikatz is used as a standalone tool for the realization of GT. There is also the possibility of the Metasploit Framework automated *meterpreter* extension as a faster solution for the execution of the attack.

5.5. Silver Ticket

Silver Ticket, introduced in 2014 along with the GT one, is another Active Directory attack related to Kerberos authentication tickets' exploitation. This technique capitalizes on a forged TGS ticket. The latter is used to authenticate and impersonate an adversary with escalated privileges for the service that represents. A malicious user may create an ST through the exploitation of a targeted host's credentials, including their account password. The ST attack has a limited scope of application compared to the GT. That is, while the first relies on a forged TGS, i.e., specifically initialized for a single service on the targeted server, the latter relies on a forged TGT that can be used as valid for the authentication of adversary hosts to any Kerberized service. Since the required hash for the execution of the ST attack is easily obtained and no communication with the domain controller is required while in use, that makes its detection harder than the GT.

In more detail, Kerberos ST is a legitimate TGS that is encrypted by the service that runs on the targeted server. Silver Ticket belongs to the general category of post-exploitation attacks, meaning that prior access and privilege escalation on the targeted system is the main prerequisite for a successful ST creation.

A high-level view of the execution of PtH and PtT in conjunction with the exploited Kerberos authentication mechanism is depicted in an UML's Activity Diagram form in Figure 4.

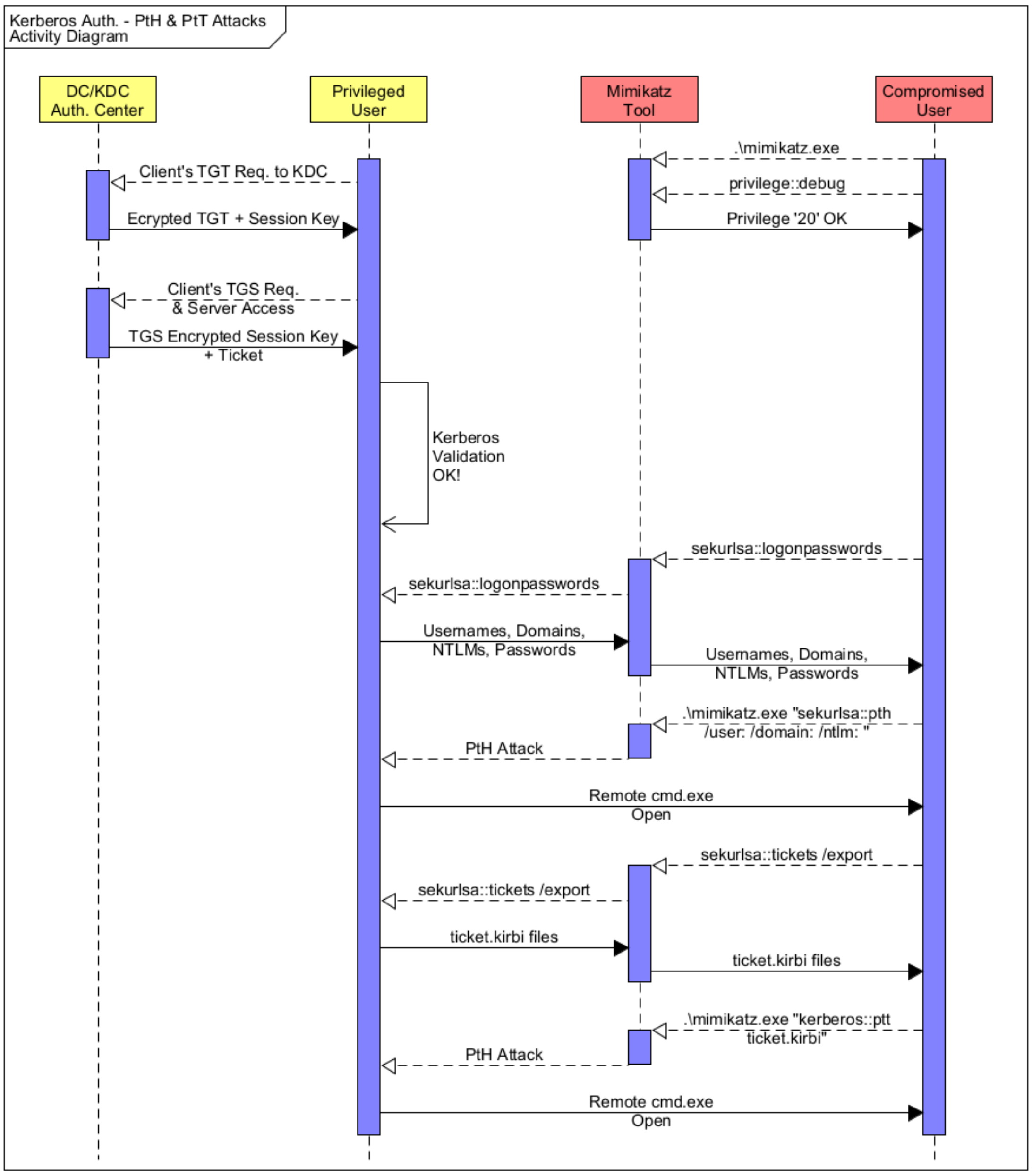

Kerberos Auth. - PtH & PtT Attacks
Activity Diagram
DC/KDC Auth. Center
Privileged User
Mimikatz Tool
Compromised User
.\mimikatz.exe
Client's TGT Req. to KDC
privilege::debug
Ecrypted TGT + Session Key
Privilege '20' OK
Client's TGS Req. & Server Access
TGS Encrypted Session Key + Ticket
Kerberos Validation OK!
sekurlsa::logonpasswords
sekurlsa::logonpasswords
Usernames, Domains, NTLMs, Passwords
Usernames, Domains, NTLMs, Passwords
.\mimikatz.exe "sekurlsa::pth /user: /domain: /ntlm: "
PtH Attack
Remote cmd.exe Open
sekurlsa::tickets /export
sekurlsa::tickets /export
ticket.kirbi files
ticket.kirbi files
.\mimikatz.exe "kerberos::ptt ticket.kirbi"
PtH Attack
Remote cmd.exe Open

Figure 4. High-level view of Kerberos authentication procedure, along with the NTLM hash extraction vulnerability, that makes KDC prone to hashing exploitation attacks, namely PtH, PtT, and Golden/Silver Ticket.

5.6. Post Exploitation on Stored Passwords with LaZagne Project

The last implemented attack pertains to the post exploitation of the already saved credentials on server's random-access memory. This was done though the open-source application named "LaZagne Project". It was performed in conjunction with the Exploitation of Remote Services attack, presented in subsection 5.1 and following the exploitation of the server's SMB file sharing protocol via Metasploit's *smb_login* auxiliary service. This a penetration testing tool which directly injects its Python code in the volatile memory of the targeted machine without leaving any trace. In this aspect, the method comprises a straightforward process for the adversary to compromise the victim's stored credentials without raising any suspicion.

6. Methodology

For the needs of this work, we relied on log files produced by Sysmon, which is both a Windows system service and a device driver. Once installed, it is resilient with all the OS activities, including reboots and event logger. Sysmon monitors and gathers in detail event-oriented information, namely process creations, network connections and modification actions, while organizing them simultaneously in folders and files of different compatible formats, namely .evtx, .xml, .csv, and .txt, to be available for further analysis.

As shown in Table 3, Sysmon supports 27 distinct types of case-sensitive events, which may be generated during logging activity of networking traffic and OS processes. The fourth column designates the EventIDs that were recognised in the context of the present work either as normal or suspicious and implemented to the proposed rule-based policy detailed in subsections 6.1, 6.2 and Appendices A.1 and A.2. The events without a check-mark are irrelevant to the conducted experiments of this paper. The rightmost column specifies the designated per attack vector and *Event ID* related subsection. The basic features and capabilities of Sysmon can be found in the tool's official website in [4].

Table 3: The 27 distinct types of Sysmon's generated events. The star exhibitor denotes normal traffic exclusively.

No.	Event ID	Description	Rules	Subsection
1	*Event ID 1*	Process creation	✓	6.1, 6.2
2	*Event ID 2*	A process changed a file creation time	✓	6.1.3, 6.1.4, 6.2
3	*Event ID 3*	Network connection	✓	6.1.4, 6.2.1, 6.2.2, 6.2.3
4	*Event ID 4*	Sysmon service state changed	✓	
5	*Event ID 5*	Process terminated	✓	6.1, 6.2
6	*Event ID 6*	Driver loaded		
7	*Event ID 7*	Image loaded		
8	*Event ID 8*	CreateRemoteThread		
9	*Event ID 9*	RawAccessRead		
10	*Event ID 10*	ProcessAccess	✓	6.2.1, 6.2.2, 6.2.3, 6.2.5
11	*Event ID 11*	FileCreate	✓	6.2.1, 6.2.2, 6.2.3
12	*Event ID 12*	RegistryEvent (Object create and delete)		
13	*Event ID 13*	RegistryEvent (Value Set)	✓	6.2.1, 6.2.2, 6.2.3
14	*Event ID 14*	RegistryEvent (Key and Value Rename)		
15	*Event ID 15*	FileCreateStreamHash		
16	*Event ID 16*	ServiceConfigurationChange		
17	*Event ID 17*	PipeEvent (Pipe Created)		
18	*Event ID 18*	PipeEvent (Pipe Connected)		
19	*Event ID 19*	WmiEvent (WmiEventFilter activity detected)		
20	*Event ID 20*	WmiEvent (WmiEventConsumer activity detected)		
21	*Event ID 21*	WmiEvent (WmiEventConsumerToFilter activity detected)		
22	*Event ID 22*	DNSEvent (DNS query)	✓	6.2.1, 6.2.2, 6.2.3
23	*Event ID 23*	FileDelete (File Delete archived)	✓	7.2*
24	*Event ID 24*	ClipboardChange (New content in the clipboard)		
25	*Event ID 25*	ProcessTampering (Process image change)		
26	*Event ID 26*	FileDeleteDetected (File Delete logged)		
27	*Event ID 255*	Event ID 255: Error	✓	7.2*

Sysmon allows the adoption of a rule-based custom configuration approach for the *config.xml* file. Each event is identified and filtered through a specified tag placed under the EventFiltering section in the *config.xml* file. The detailed list of all the available tags along with the list of the conditional statements, which are necessary for the initialization of the *config.xml* file, are available in the tool's official documentation [4].

Precisely, within the context of this paper, Sysmon was used to filter and identify events related to the executed LM attacks included in Section 5. Specifically to the attacks, due to their nature, they were divided in two main categories, namely *Exploitation of Remote Services and Credential Exploitation Attacks* and discussed in subsections 6.1 and 6.2, respectively. Listing 1 presents the necessary initialization statements for the *config.xml* file to be loaded in Sysmon's configuration. As seen in the first line of the listing, the declaration of the *schemaversion*, under the tag <*Sysmon schemaversion="13.30"*>, is fundamental for the initialization of the *config.xml* file. Moreover, the declaration of the utilized *hashalgorithms* as well as the <EventFiltering> tag, which states that the nature of the configuration file has to do with the filtering of Sysmon Events, are both of equal importance. The final required statement of xml code is the specification of the necessary for each concept *RuleGroup*. In our case, three different RuleGroup(s) have been created, one for each category of EventFiltering, as follows:
<RuleGroup name="Sysmon_Event_1(ProcessCreate)" groupRelation="or">
<RuleGroup name="Sysmon_Event_3(NetworkConnect)" groupRelation="or">
<RuleGroup name="Sysmon_Event_5(ProcessTerminate)" groupRelation="or">

The *groupRelation* was set in purpose to the *"or"* value for each *RuleGroup* to be iterated during an EventFiltering of a *.evtx* Sysmon file, as presented in Listing 1.

```
<Sysmon schemaversion="13.30">
  <HashAlgorithms>md5,sha256,IMPHASH</HashAlgorithms>
  <CheckRevocation/>
  <EventFiltering>
  <RuleGroup name="Sysmon_Event_1(ProcessCreate)" groupRelation="or">
```

Listing 1: Sysmon config.xml initial statements

As seen in Listing 2, the *onmatch* statement was applied for the identification of the events that match the rule, while the *"include"* or *"exclude"* values means that only matched events are filtered or not, respectively.

For the purposes of the configuration of the *config.xml* file, with the proposed rule-based methodology, more than 150 rules have been created. The rationale behind the creation of these rules is detailed in subsections 6.1 and 6.2. The current work relies on the analysis of the particular characteristics associated with each attack separately and in combination with the observations of network traffic during the execution of each attack given in Section 5. The experiments continued with the collection and analysis of the DLL files; these were loaded as legitimate from the malicious tools with which the LM experiments were executed. The results of the analysis of the DLL files related to the malicious tools with which the LM of Section 5 were executed, is integrated to the proposed rule-based calibration of the Sysmon tool. The final result, is a novel EDR solution based on Sysmon. Simply put, our proposed solution particularly focuses on the generic identification of potential malicious LM by means of legitimate logging activity.

According to the related work in [10,20,21], the utilization of an EDR system that is calibrated with custom rules to filter malicious processes based on their names, hashes, and network characteristics, is proposed as a necessity towards the avoidance of modern and sophisticated LM. However, this approach does not guarantee the effective identification of the very special characteristics that many malicious tools enclose, ending up evading their successful recognition, either through being renamed or rebuilt by the opponent. A solution to the problem of the identification and detection of malicious tools is proposed by researchers in [10–12] through the incorporation of DLL analysis and the careful study of their unique per-tool characteristics.

```
<ProcessCreate onmatch="include">
  <CommandLine condition="contains">C:\Windows\system32\conhost.exe 0xffffffff -ForceV1</CommandLine>
  <ParentCommandLine condition="is">C:\Windows\system32\wevtutil.exe install-manifest </ParentCommandLine>
  <Image name="conhost" condition="is">C:\Windows\System32\conhost.exe</Image>
  <Image name="conhost" condition="end with">.exe</Image>
  <ParentImage condition="is">C:\Windows\System32\cmd.exe</ParentImage>
  <ParentImage condition="is">C:\Windows\System32\wevtutil.exe</ParentImage>
  <CurrentDirectory condition="is">C:\Windows\</CurrentDirectory>
  <Description condition="is">Console Window Host</Description>
  <!--Console Window Host-->
</ProcessCreate>
```

Listing 2: conhost.exe "include" rule related to the executed tickets attacks

6.1. Exploitation of Remote Services

Four variations of this attack were executed and presented below. In all four variants, the final goal of the attacker was the execution of remote commands via a PowerShell terminal. For the feature extraction needs, which will be related to the finally proposed rule-based EDR policy, 30,000 Sysmon logs were collected, representing 1 hour of the described attack. Precisely, each sub-category of the T1210 technique [1] was performed consecutively for 10 min each, not exceeding one hour in total.

From an attacker's perspective, access can be granted in multiple ways towards the LM with stolen credentials over a Windows Active Directory environment. Specifically, several features were identified as potentially malicious related with Windows Services: (i) Service Control Manager (SCM), which pertains to the creation or starting of a Windows Service as remote, (ii) Remote Manager (WinRM), which refers to the remote invocation of admin privileges on services, (iii) RDP, referring to the process engagement remotely through RDP session, (iv) Management Instrumentation (WMI), pertaining to the arbitrary code creation/execution via a *Win32_Process.Create* on a remote host, (v) Distributed Component Object Model (DCOM), concerning objects invoked remotely through PowerShell, and (vi) Task Scheduler, referring to the remote execution or creation of scheduled tasks. On top of everything else, the so-called *Bluekeep* vulnerability (CVE-2019-070) was exploited as an evidence of the effectiveness of the proposed rule-based analysis. The three distinct variations of the remote services attack plus the extra fourth, along with the discussion per subject, are presented in more detail as follows.

6.1.1. Exploitation of ms17-010

This was conducted successfully against a vulnerable MS Windows server with the aid of Metasploit. The success of the previous placement is proved by the appearance of the command line with the following message: "windows/smb/ms17_010_eternalblue >...". It is worth mentioning that the targeted server responded with a successfully activated *Meterpreter* session of the EternalBlue vulnerability after several repeated attempts (at least ten). The attack is deemed successful when PowerShell admin rights are available for the execution of core Windows commands and LM expansion to other hosts. For instance, the *whoami* command was initially executed to prove the admin rights of the attacker's on the target and *net user* to acquire all relevant to the targeted domain user accounts.

Discussion: With reference to malicious endpoint command abuse, the analysis revealed Process Creation i.e., *EventID 1 with reference to Table 3*, vector as the most prone to exploitation from opponents. Specifically, more than 25% or 7,500 logs were related with the creation of processes on either the source or destination hosts. Command line administrative tools were among the most recognised on the logs related with the attacker's machine, namely *sc.exe*, *schtasks.exe*, *winrs.exe*, *PowerShell*, *cmd.exe* and many others. On the other hand, among the most identified parent processes linked to the aforementioned potentially malicious Windows Services features, were *services.exe (SCM)*, *svchost.exe (Scheduled Tasks)*, *wmiprsve.exe (WMI)*, *mmc.exe (DCOM)*, *wsmprovhost.exe (WinRM)*, and *explorer.exe (RDP)*. Creating a rule-based Sysmon policy related to the above-mentioned processes on a remote endpoint may reveal compromised *PowerShell* services strongly related to the execution of LM attacks and arbitrary code injection. Besides, the existence of *svchost.exe* and *mmc.exe* process features in the *Sysmon* logging activity may also be a distinctive indicator for remote execution of *Impacket Python scripts*. That is, *Impacket* Python classes are used for Microsoft's network protocols implementation and are popular among threat actors. They are examined as potential to be exploited for LM purposes since the adversaries may leverage the many parallel *svchost.exe* running services on a legitimate host to conceal their presence.

6.1.2. Exploitation of EternalBlue

This attack was performed via Metasploit's *smb_login* auxiliary module and *psexec*, revealing the targeted server of Figure 2 as prone to the *smb_login* credential leakage and execution of remote commands vulnerability, namely "exploit/windows/smb/psexec (Microsoft Windows Authenticated User Code Execution)"). A *Meterpreter* session is also necessary to be activated for the attack to be considered successful. To verify that the compromised system was indeed the Windows server the *systeminfo* command was executed via an already opened cmd admin terminal on the server's side with the "execute -f cms.exe -I -H" command.

Discussion: With respect to *EternalBlue* vulnerability via *smb_login* and *PsExec*, the conducted experiments revealed leveraging of Windows SMB as potentially malicious. Specifically, SMB access with administrative privileges, namely *admin$*, *ipc$* and *c$*, is considered a suspicious behaviour which implies the manipulation of the Windows system's binaries. This behaviour was identified within the collected logs, during the execution of the Metasploit's *smb_login* module and *PsExec* tool and prior to the execution of remote services through *Meterpreter* session on the targeted host. In the context of this work, we utilized the *net.exe* service to connect via Metaspoilt to the SMB protocol's sharing environment towards privilege escalation. SMB share's enumeration followed, through the mapping of the server's *smb-admin c$*, generating a number of interesting *Sysmon* log entries on the targeted host. To begin with, since mapping of the *c$ admin shares* within the borders of a SOHO or corporate network do not constitute legitimate user's normal activity, any event generated on Sysmon with *EventID 1* related to the *net.exe*, should be identified as potentially malicious. Moreover, logs related to the *Image*, *CommandLine*, *CurrentDirectory*, *Hashes*, *ParentImage* and *ParentCommandLine* of *net.exe* activity, should also be integrated to the proposed rule-based Sysmon policy. Similar to the age of *net.exe*, a *ProcessCreate* rule for the *whoami.exe* was also included. Despite being a Windows native executable, it is uncommon for a legitimate user with basic credential access to run it through PowerShell [22]. This pattern was observed within the collected logs and was integrated as a proactive threat-hunting measure and not as a rule that could serve as standalone.

6.1.3. Deployment of WannaCry

In addition to the two aforementioned versions of the *"Exploitation of Remote Services"* attack, the well-known *WannaCry* ransomware was propagated deliberately to the VMs of the testbed because of the exploitation of ms17-010, and the execution of arbitrary remote commands and code. To assist the expansion of the malware, server's UAC was disabled remotely with the use of the *smbexec* 2.0 tool. The latter is a post exploitation tool used in a similar way with *psexec*. Access to any kind of credentials is mandatory beforehand for acquiring local or domain account credential rights. Among the plethora of utilities that come embedded with *smbexec*, such as the enumeration or exploitation of a targeted system and the stealing of hashes, disabling UAC is particularly useful for malware establishment on a network host. The status of the target's UAC was checked prior to and after the execution of the disabling tool to verify the successful disabling of the specified service. WannaCry was downloaded from GitHub and used to infect the Windows Server 2019 VM. After the exploitation of the EternalBlue vulnerability, WannaCry encrypted every file in the targeted server, displaying a message demanding payment in Bitcoin. At the same time, *Wireshark* has been used to capture the network activity. As expected, after the VM's infection, a mass packet exchange started, demonstrating unknown to the proposed testBed IP addresses, such as 89.163.210.241 and 34.107.221.82.

Discussion: Regarding the *WannaCry* spread on the targeted server, *"ProcessCreation"* and *"ProcessChanged"* events *(EventIDs 1 & 2)* were identified as the most relevant to the target's infection, representing more than 40% (or 12,000 logs) of the logging activity during the execution of the experiments. Specifically, both events are generated when a *Process* alters a file and do not constitute an indication of malicious activity when used in isolation. However, when combined with a number of factors, namely *CommandLine*, *CurrentDirectory*, *Hashes*, *ParentImage*, *Parent* and *CommandLine* rules, they form an effective data-driven proactive detection mechanism towards the identification of *WannaCry* expansion. With reference to the identified processes after the deliberate execution of *WannaCry*, *Wcry.exe* and *tasksche.exe* were isolated as the most relevant logs to the attack, along with their *path* and *ProcessID 3024*. On the other hand, the execution of the *Wcry.exe* process, generated a number of interrelated *"r.wnry"* executables. Precisely, those files are generated during the decryption of the infected files and after the ransom's payment in bitcoin through the *"s.wnry"* file. The *"taskse.exe"* process was identified as a *ProcessCreation* log that was also created by *Wcry.exe* for reasons of Windows RDP sessions enumeration, whereas *"taskdl.exe"* was also created for file encryption with the *".WNCRYT"* extension. The creation of the *"t.wnry"* and *"u.wnry"* files was also identified as generated by *Wcry.exe* for reasons of AES encryption and decryption, respectively. Besides, the continuous display of the *"C://Users//...//Desktop//Eula.txt.WNCRYT"* objects denotes file encryption with the aforesaid *"taskdl.exe"* process.

Malware spreading behaviour produced two more log categories, namely "attrib.exe" and "icacls.exe", which are related to commands execution of altering the attributes of the targeted files. Moreover, each process executed the "attrib +h" and "icacls . /grant Everyone:F /T /C /Q" commands, respectively. The former is related with malware's tendency to hide the infected files, whereas the latter grants administrative access to all the directories and sub-directories. Regarding the above-mentioned behavior, 12,000 (or ≈40%) "ProcessCreation" and "ProcessChange" Sysmon's event-logs, 3258 (or ≈11%) were identified as exclusively related to the WannaCry infection.

6.1.4. Exploitation of BlueKeep

To extend the conducted research, and as a proof of concept to the generic nature of the proposed data-driven methodology for the identification of Exploitation of Remote Services attacks, the server (Figure 2 and Table 2) was exposed to the BlueKeep (CVE-2019-0708) RCE vulnerability. BlueKeep is related to the RDP and RDS, revealing a similar behaviour to the above-mentioned WannaCry exploitation of the EternalBlue vulnerability. The exploitation was conducted via Metasploit's CVE-2019-0708 dedicated module. At this point, the inability of the aforesaid module to provide automatic vulnerability analysis to the targeted system, should be mentioned as the reason for the manual configuration of all the relevant technical aspects of the targeted server, namely IP address, username, and password.

Discussion: With respect to the proposed rule-based Sysmon policy, 689 (or ≈4%) *"ProcessCreation"* and *"Process-Changed"* related logs *(EventIDs 1 & 2)* were identified as strongly correlated to suspicious LM behaviour on RDP. The examined rules were exported during the first stage of the experiments presented in subsection 6.1. Despite promising results, the in-depth reexamination of the collected logs revealed the absence of *"NetworkConnection"* logging activity with *"EventID 3"*, specialized enough to describe the core functions of the RDP and RDS Windows protocol and services, respectively. Sysmon was updated with rules, generic enough to alert in case of RDP BlueKeep exploitation. Simply put, Windows services which utilize the same port as RDP protocol, namely *Destination Port 3389*, where identified and imported as rules (*mstsc.exe*, *RTSApp.exe*, *ws_TunnelService.exe*, *RemoteDesktopManagerFree.exe*, *RemoteDesktopManager.exe*, *RemoteDesktopManager64.exe*, *mRemote.exe*, *Terminals.exe*, *spiceworks-finder.exe*, *chrome.exe*, *thor.exe*, *thor64.exe)*. The complete description of the selected rules is presented as follows:

- *mstsc.exe*: It is related to Microsoft Terminal Services Client (MSTSC) Windows shared library (DLL) and allows access to RDP clients to perform a remote desktop connection through command line or PowerShell. If leveraged by an adversary, the relevant DLL may allow the execution of arbitrary controversial code remotely without the mediation of GUI or a SOCKS proxy connection [23].
- *RTSApp.exe*: It is related with the Royal TS remote management software from *code4ward*. Royal TS supports both Windows RDP and RDS, protocol and services. If not found in `C:\\ProgramFiles\\code4ward\\royal_ts`, then it should be checked for potential malware activity which refers to LM attack [24,25].

- *ws_TunnelService.exe*: This process belongs to VMware and there are no references to Virustotal for potential malicious behaviour. However, when identified in a system that neglects VMware from its utilities, then the collected digital signature's hash algorithm of the file should be checked as potentially malicious [26].
- *RemoteDesktopManagerFree.exe*, *RemoteDesktopManager.exe*, *RemoteDesktopManager64.exe*, *mRemote.exe*: The mentioned processes are strongly related to Windows handling of remote desktop connections. They comprise nonstandard tools of Windows that utilize destination port 3389 and should be considered as potentially dangerous for LM orchestration [25,27,28].
- *Terminals.exe*, *spiceworks-finder.exe*, *thor.exe*, *thor64.exe*: These executable files are no Windows native and when in existence should be evaluated as potentially malicious and related to malware-based LM behaviour [24].
- *reg.exe*, *chrome.exe*: The second process is one of the most common among Windows users, as it is the executable of the Chrome browser. Despite the reliability of the process, adversaries can take advantage of the frequency with which it appears on the target system by leveraging Chrome's launcher. This is accomplished through the manipulation of the target's registry permissions remotely to grant RCE and RDP exploitation eventually. The simultaneous presence of the two processes or the occurrence of the second in a subfolder of the user's profile folder, in conjunction with the utilization of TCP port 3389 should be examined as malicious [29,30].

Regarding the aforesaid Sysmon's networking logging activity, 456 “Network Connection” EventID 3 logs were identified with more than half of them (or ≈242 Sysmon logs) related to the executed LM attacks. For easy reference, Table 10 in Appendix A.1 summarizes the most important rule-based features of this Section 6.1 as they emerged from the four executed experimental variants of the Exploitation of Remote Services attack.

6.2. Credential Exploitation Attacks

With reference to credential exploitation techniques, five distinct attacks were executed, namely PtH, PtT, GT, ST and Post Exploitation on Stored Passwords with LZP. For the needs of the rule-based analysis presented in this work 60,000 Sysmon logs were collected, representing 60 min of continuous execution of each credential exploitation respectively. Precisely, for demonstrating as clearly as possible the distinct characteristics of the current experiments, each sub-category of Mitre's T1550 Technique [1] was performed consecutively for 60 min each and not exceeding the 240 min in total.

As already noted, for executing the experiments, we relied on the *Mimikatz* tool. This choice was driven by factors such as the tool's popularity among adversaries, the multiple versions available (*GitHub legacy edition*, *PowerShell Invoke-mimikatz*, ransomware such as *BadRabbit* or *NotPetya)* and the possibility to execute the tool with *Metasploit framework*. *Mimikatz* comprises various modules, the most significant of which are presented in Table 4:

Table 4: Most recognised features of Mimikatz tool used during the experiments

Feature	**Description**
privilege module	It is activated when the *“mimikatz # privilege::debug”* is executed allowing access to a potentially adversarial user to debug a system's process that otherwise is restricted.
sekurlsa	It is activated when the *“log sekurlsa.log”* is executed and is followed by an *“OK”* confirmation on the terminal. It is responsible for the extraction of any credential stored on Windows LSASS process (through *“mimikatz # sekurlsa::logonpasswords”* command), namely usernames, passwords, NTLM hashes, domain, etc.
kerberos	This process do not require any *“debug”* administrative privilege to be granted beforehand. It is used in PtT, GT, and ST attacks towards the creation of long-term TGT tickets, which permit adversaries being identified as legit.
lsadump	It allows access to *Windows Security Account Managers (SAM)* database, where usernames, passwords, and NTLM - LM hashes can be extracted. It is an alternative to the *sekurlsa* module.

6.2.1. Pass the Hash

As already mentioned in Subsection 5.2, PtH is a credential theft technique and LM attack at the same time. PtH was executed against client 3 of Table 2. The targeted Windows 10 system stored the passwords in NTLM hashing form. Mimikatz was chosen for the execution not only of *PtH* attack, but also for the *PtT* and *GT/ST* ones. The related account's, with username *chr.smilio*, hashing passwords were extracted from the server's memory and the Windows LSASS.exe service, through *Mimikatz's* administrative command line remote execution. To acquire admin rights the *“Privilege “20” OK”* message is necessary to be revealed in command line for the PtH command to be successfully executed

(*"./mimikatz.exe "sekurlsa::pth /user:chr.smilio /domain:SYSMON_SET /ntlm: eed224b4784bb040aab50b8856fe9f02""*). Finally, a new compromised command line was opened denoting the success of the whole process.

Mimikatz extracted the NTLM hashing strings in a few minutes and cracked the targeted system's encrypted passwords in plaintext with brute-force techniques. The reader should keep in mind that besides *Mimikatz* there are many others also well-known methods and tools to achieve NTLM extraction and password cracking. Among the most popular tools, *PWDump7*, *PWDumpX* and *Quarks PWDump* are used for displaying and acquiring password hashes from local and remote accounts, as well as cashed passwords. *PowerMemory (RWMC Tool)* is another effective cmd executed tool for extracting credential information stored in memory or hard-drives. *ProcDump* is also used to extract LSASS authentication information, which is later processed offline via Mimikatz.

6.2.2. Pass the Ticket

As mentioned in Section 5.3, PtT is similar to PtH regarding Mimikatz's execution. The major difference between the two attacks is the duration of the access that is granted on the compromised host. PtT has limited exploitation timeline, with Kerberos TGTs expiring after 10 hours of usage. On the other hand, PtH has unlimited timeline as it is related to hashes that do not change in a frequent basis. PtT was executed with the same Mimikatz methodology that is presented for PtH in subsection 5.2 up to the point of the successful acquisition of administrative privileges. All the Kerberos tickets were extracted with the *".kirbi"* extension in order to be used to the final exploitation of the Kerberos ticket client-server authentication procedure. Finally, the extracted ticket was *"passed"* with Mimikatz (*"./mimikatz.exe "kerberos::ptt ticket.kirbi"*) towards the acquisition of remote access to the targeted server's *"/192.168.1.7/admin$"* folder.

6.2.3. Golden/Silver Ticket

As it concerns the GT attack, Mimikatz supports its execution toward the successful creation of a *"golden"* Kerberos ticket. For the needs of this work, and with reference to Figure 2 and Table 2, the *SYSMON_SET* domain's *"chr.smilio"* account of Client STA 3 was leveraged and exploited. As mentioned in subsections 5.4 and 5.5, this set of attacks is a combination of the already presented PtH and PtT attacks. For the creation of a GT to be successful, the exploitation of an extracted NTLM *"Krbtgt"* hash is a requirement, along with the domain's *Name*, the *SID*, and the *Username* of the targeted host.

These four elements are combined with Mimikatz to successfully create the *"golden ticket"* (*"kerberos::golden /user:krbtgt /domain:SYSMON_SET /sid:S-1-5-21-902028059-194221605-2102066478 /krbtgt:d125e4f69c851529045ec95ca80fa37e /ticket: krbtgt.tck /ptt"*) and allow the adversary to move laterally. All the submitted tickets after the execution of this attack were retrieved with the *"kerberos::list"* command. After the generation of the NTLM Hash for the *"Krbtgt"* account, Kerberos will perceive this TGT as trusted. Therefore, any owner of this ticket, either legitimate or malicious, will be provided with administrative privileges for unlimited access to network facilities related with Kerberos authentication, including the Domain Controller. It should be noted that the same procedure was followed during the creation of a forged TGS ticket and the exploitation of the related service. The only difference is in the use of the *PtT* parameter (*"/ptt"*) during the ticket's production process to indicate the use of the *PtT* technique to produce the Silver ticket (*"kerberos::golden /domain:SYSMON_SET /sid:S-1-5-21-902028059-194221605-2102066478 /target:WIN-J23NIGGP1Q6.sysmon_set.lobal /service:cifs /rc4: 1d86942baf284a38c97b63025fc8ccb8 /user: Administrator /ptt"*).

6.2.4. Discussion upon Mimikatz-related Attacks

From an attacker's perspective, during PtH execution, a recognizable number of more than 1,500 (or 2.5% of the total log file) *ProcessAccess* event logs (*EventID 10*) were identified as the most pertinent to Mimikatz-related attacks of subsection 6.2. Specifically, the aforesaid activity is related to the unauthorised access in *Windows LSASS* process by the adversary with Mimikatz (as in this paper) or any other dumping tool of password and hashes. As shown in the fourth column of Table 11, the log activity analysis revealed a significant number of Sysmon's *"EventFields"* related to *EventID 10* events. Specifically, *lsass.exe* (*"EventField: TargetImage"*) executable was the most prominent with 600 out of 1,500 identified logs due to the repeated execution of the attack and exploitation of *LSASS* process through *NTLM hash dumping*.

It is important to note that the identified activity of *LSASS* accessing was executed in parallel with an equal number of elevated privileges on access granting events, with *0x1010* and *0x1410* identifiers (*"EventField: GrantedAccess"*). The limited presence of three more *"GrantedAccess"* identifiers was also perceived, namely *0x1438(a)*, *0x1FFFFF* and *0x143a*, all scattered throughout 50 *EventID 10* events out of the 600 logs. Above that, a number of ≈600 *mimikatz.exe* executable processes (*"EventField: SourceImage"*) with *"TargetProcessId"* equal to 492 were identified in combination with ≈200 powershell.exe and cmd.exe instances. Note that the aforesaid features in isolation do not guarantee the identification of PtH, however in combination they could be considered as the springboard of the overall process of unveiling the *"sekurlsa"* module of the mimikatz-related credential theft techniques presented in this section.

Overall, we consider that the application of the hitherto recognized *PtH-related* rule-based features is fully compatible to be extended in *PtT*, *GT*, and *ST* attacks. This conclusion emerges from the study of each attack's functionality presented in this Section. Specifically, *PtH* is based on the successful extraction of *NTLM hashes*, which are used via the *"sekurlsa::pth"* module of Mimikatz to pass the hash and acquire a legit and stealth access to the targeted system. On the other hand, PtT is strongly related to the extraction of TGT tickets in *.kirbi* files, and their utilization to authenticate as a legitimate user. As in PtH, PtT also leverages Mimikatz's *"sekurlsa"* module to pass the ticket to the target. Moreover, the GT and ST techniques comprise both NTLM hash dumping techniques and TGT ticket forging towards the creation of the migrated Golden and Silver passed tickets. Summing up, in terms of the common behavior and operation of the three aforementioned attacks, the rule-based policy proposed for the PtH attack finds scope in them as well.

Despite the vast existence of *ProcessAccess* events (*EventID 10*) related to the abuse of the *lsass.exe* process, another 3,200 events with *EventIDs 1* (*ProcessCreation*), *ID 5* (*ProcessTermination*) and *ID 11* (*FileCreate*) were stood out in the collected Sysmon's logs. Precisely, the events with *IDs 1* and *5* were exclusively related to Mimikatz's execution to the targeted system, either via *psexec* or *Metasploit*. On the other hand, events described with the *"FileCreate"* rule flag, were captured by Sysmon during the extraction with Mimikatz of all the Kerberos tickets into a specified folder to be used for the final exploitation of Kerberos tickets in PtT, GT, and ST attacks. The identification of the *.kirbi* extension in the filenames (*EventField: TargetFilename*) of the extracted files is the most obvious indication of the existence of tickets related to the Kerberos protocol. Among the aforesaid 3,200 logs, the majority (or ≈2,400) was flagged by the *"TargetFilename"* rule. Besides, through Sysmon's log traffic observation, and due to the nature of the three executed Kerberos tickets attacks, the identified number of ≈360 *EventID 11* (*FileCreate*) can be considered a strong indicator of the Kerberos *TGT* tickets exploitation into *.kirbi* extension files.

```
<!-- Local Security Authority Process and Registry Console Tool are monitored with the use of the aforementioned ProcessCreate rules, acting as the main axis of our methodology towards the identification of Lateral Movement Privilege Escalation and Persistence Attacks. -->

     <ProcessCreate onmatch="include">
        <CommandLine condition="contains">C:\Windows\system32\lsass.exe</CommandLine> <!-- Possible indication for the execution of the Lateral Movement Skeleton Key Attack -->
        <CommandLine condition="contains">C:\Windows\system32\svchost.exe -k RPCSS -p</CommandLine>
        <CommandLine condition="contains"> C:\Windows\system32\svchost.exe -k netsvcs -p -s wuauserv</CommandLine>
        <CommandLine condition="contains">C:\Windows\system32\reg.exe query hklm\software\microsoft\windows\softwareinventorylogging /v collectionstate /reg:64</CommandLine> <!-- Indication of a potential attemp to alterate registry regords. -->
        <CommandLine condition="contains">C:\Windows\system32\svchost.exe -k UnistackSvcGroup -s CDPUserSvc</CommandLine>
        <ParentCommandLine condition="is">C:\Windows\system32\cmd.exe /c C:\Windows\system32\reg.exe query hklm\software\microsoft\windows\softwareinventorylogging /v collectionstate /reg:64</ParentCommandLine> <!-- Indication of a potential attemp to alterate registry regords. -->
        <Image name="lsass ProcessCreate" condition="is">C:\Windows\system32\lsass.exe</Image>
        <Image name="svchost ProcessCreate" condition="is">C:\Windows\System32\svchost.exe</Image>
        <Image name="ProcessCreate" condition="end with">.exe</Image>
        <Image name="reg64 ProcessCreate" condition="is">C:\Windows\System32\reg.exe</Image>
        <ProcessId condition="is">668</ProcessId>
        <ProcessId condition="is">960</ProcessId>
        <ParentImage condition="is">C:\Windows\System32\wininit.exe</ParentImage>
        <ParentImage condition="is">C:\Windows\System32\services.exe</ParentImage>
        <CurrentDirectory condition="is">C:\Windows\System32\wininit.exe</CurrentDirectory>
        <Description condition="is">Local Security Authority Process</Description>
        <Description condition="is">Registry Console Tool</Description>
        <!--Local Security Authority Process, Registry Console Tool-->
```

Listing 3: lsass.exe "include" rule related to the executed tickets attacks

Moreover, the presented research was extended to identify extra features for each specified attack vector in isolation. Remotely executable tools were identified as applicable to the process of compromising the targeted server and the successful acquisition of administrative rights, namely *klist.exe*, *mimikatz.exe*, *lazagne.exe*, *psexec.exe*, *smb.exe*, as shown in Listing 3. These tools are strongly related to the extraction of *NTLM hashes* and *TGT tickets*, which follow the privilege escalation phase. Moreover, with reference to Listing 4, *klist.exe ProcessCreate* was included as one of the most obvious proofs of the execution of Kerberos tickets related to LM, namely PtH, PtT, GT, and ST. Specifically, *"klist.exe"* is a Windows command line executable which prints a list with the most recently cached Kerberos tickets. It should be noted that for *"klist"* as for most aforesaid executable, administrative privileges come as a prerequisite.

```
<!-- The ProcessCreate rule with Description "Tool for managing the Kerberos ticket cache" was included as one of the
     most obvious proofs of the execution of Kerberos
tickets related Lateral Movement attacks, such as Pass the Hash, Pass the Ticket, Golden Ticket, Silver Ticket. -->

    <ProcessCreate onmatch="include">
      <ProcessId condition="is">4196</ProcessId>
      <Description condition="is">Tool for managing the Kerberos ticket cache</Description>
      <CommandLine condition="is">klist</CommandLine>
      <CurrentDirectory condition="contains">C:\Users\Administrator\</CurrentDirectory>
      <Image name="klist" condition="is">C:\Windows\System32\klist.exe</Image>
      <ParentImage condition="is">C:\Windows\System32\cmd.exe</ParentImage>
      <ParentCommandLine condition="is">C:\Windows\system32\cmd.exe</ParentCommandLine>
    </ProcessCreate>
```

Listing 4: klist.exe Kerberos Tickets list

As it concerns privilege escalation techniques, *EventID 1 ProcessCreate* rules were the most extensively identified during the execution of LM. Precisely, the *CommandLine EventFiels* presented in Listing 5, comprise the identification of *keylogging* or *keyboard capturing* (*"Get-Keystrokes"*), the capturing of screenshots from the targeted machine (*"Get-TimedScreenshot"*), and the extraction of cleartext credential Windows objects (*"Get-VaultCredential"*, *"Invoke-CredentialInjection"*). Further, the aforesaid executables, refer to *PowerShell* Windows modules which can be executed during the target's enumeration and privilege escalation phases, within the context of a penetration testing or a real-life LM.

Besides, the possibility of the execution of credential extraction and manipulation attacks via *PsExec* as an escalation to one of the aforementioned exploitation of remote services operations, revealed the importance of *EventID 3 NetworkConnection* and *EventID 22 DNSEvent* (*DNS query*) events. Regarding the former, 150 logs were collected as related to successful remote connections via the RDP protocol [*"C:/Windows/PSEXESVC.exe" (EventField: Image)"*] and also associated with the source and destination IP addresses of the attacker's machine and the targeted server host, respectively (SourceIp: 192.168.1.11, DestinationIp: 192.168.1.8, as presented in Table 2). Regarding *DNSEvents*, 80 logs were collected by *Sysmon* with *"EventField: Image:.../PSEXEC.exe"* and *"QueryName: sysmon_set"* as the domain's name, all revealing potential domain exfiltration for stored hashes and passwords with Mimikatz.

```
<ProcessCreate onmatch="include">
        <CommandLine name="Privilege Escalation" condition="contains">Get-Keystrokes</CommandLine> <!-- Keystroke
            logging, often referred to as keylogging or keyboard capturing, is the action of logging keys pressed,
            time and the active window. -->
        <CommandLine name="Privilege Escalation" condition="contains">Get-TimedScreenshot</CommandLine> <!-- Takes
            screenshots and saves them to a folder, with a timestamp to reveal the time and date of the screenshot.
            -->
        <CommandLine name="Privilege Escalation" condition="contains">Get-VaultCredential</CommandLine> <!-- Displays
            Windows vault credential objects including cleartext web credentials. -->
        <CommandLine name="Privilege Escalation" condition="contains">Invoke-CredentialInjection</CommandLine> <!--
            Displays Windows vault credential objects including cleartext web credentials. -->
        <CommandLine name="Privilege Escalation" condition="contains">mimikatz</CommandLine> <!-- Mimikatz is an open-
            source application that allows users to view and save authentication credentials like Kerberos tickets.
            -->
    <!-- <ProcessCreate onmatch="exclude"> -->
```

Listing 5: Privilege Escalation ProcessCreate rules

To make the collected features as generic as possible, two more interventions were implemented in the targeted server. The former is related to the activation of the *"WDigest"* authentication and the latter with enabling *"Manages Microsoft Base Smart Card Crypto"* capability. As it concerns *WDigest*, if enabled, it forces *lsass.exe* to store on system's memory user-related passwords in plaintext form. On the other hand, when *Smart Card Crypto* capability is enabled it makes *LSASS* to store the dedicated to each user NT hash along with the unique key of the card itself. Both events were captured by *Sysmon* as *EventID 13* (*RegistryEvent (Value Set)*), with *"Image: C://WINDOWS//regedit.exe"* and *"TargetObject: HKLM/SYSTEM/CurrentControlSet/Control/SecurityProviders/WDigest"*, *"TargetObject: HKLM/SOFTWARE/Microsoft/Cryptography/Defaults/Provider/Microsoft Base Smart Card Crypto Provider"*, respectively.

Many rules have also been *excluded*, meaning that they were filtered continuously until their rule was matched. Most of the times these are common Windows OS processes that should be excluded to avoid causing unwanted noise to the Sysmon event filtering. A hint of the rule that were set as excluded is depicted in Listing 6.

```
<ProcessCreate onmatch="exclude">
  <!--COMMENT: Exclude mostly-safe sources and log anything else.-->
    <Image name="wininit" condition="is">C:\Windows\system32\wininit.exe</Image> <!-- The genuine wininit.exe file is
         a software component of Microsoft Windows Operating System by Microsoft Corporation. -->
    <Image name="csrss" condition="is">C:\Windows\system32\csrss.exe</Image> <!-- Csrss.exe (also known as Client
        Service Runtime Process) is a legitimate and important process that runs in Windows Operating Systems. -->
    <Image name="winlogon" condition="is">C:\Windows\system32\winlogon.exe</Image> <!-- Very important part of the
        Windows operating system, and Windows will be unusable without it. -->
</ProcessCreate>
```

Listing 6: Excluded common Windows processes

Mimikatz's DLL Analysis: In addition to the *rule-based EDR* policy proposed, log analysis was taken further to the identification of the DDL information that was transferred during Mimikatz execution. The captured DLLs were included within the collected 60,000 Sysmon logs that were taken during the extraction of the NTLM hashes, the exploitation of PtH, PtT, GT, and ST attacks. According to the authors in [10], even thought adversaries tend to rename or even rebuilt malicious applications like Mimikatz, the existence of legitimate DLLs which are loaded together with the aforesaid tool may give great insight regarding the uniqueness of the DLL relationship between distinct editions of Mimikatz and Windows OS, leading eventually to a robust and effective proactive identification of LM.

Table 5: DLL list for Mimikatz (Compared to Matsuda et al. [10])

DLL list for Mimikatz (Compared to Matsuda et al. [10])		
No.	**Identified DLL**	**Related Work**
1	cryptbase.dll	✓
2	cryptdll.dll	
3	hid.dll	
4	imm32.dll.dll	✓
5	kernel32.dll	✓
6	msctf.dll	✓
7	ntdll.dll	✓
8	samlib.dll	
9	sechost.dll	✓
10	shell32.dll	✓
11	user32.dll	✓
12	vaultcli.dll	
13	wininet.dll	✓
14	WinSCard.dll	

Specifically, 3,500 *DLL files* (≈ 6% of the total log record corpus) were identified over the 60,000 logs, however only 460 of them (≈ 13%) were related to *Mimikatz*. Fourteen different DLL information files were identified and presented in Table 5, whereas only 9 out of 14 (64%) are referenced by the so far related work [10]. We argue that future work on the subject of EDR, will evolve the presented rule-based policy and list of the *Mimikatz* related DLL files into a signature-driven deep learning driven IDS solution. Tables 10 and 11 summarize the most important rule-based features of subsections 6.1 and 6.2 as they emerged from the conducted experiments and the extended DLL analysis.

6.2.5. Post Exploitation on Stored Passwords with LaZagne Project

The execution of this attack started at the end of the previously presented exploitation of the *smb_login* vulnerability of the Windows Server with Metasploit and the successful acquisition of the meterpreter session on the targeted host, presented in subsection 6.1. The *lazange.exe* tool was remotely uploaded and executed on the targeted host's *"/Downloads"* folder through the command line utility with administrative rights. After several attempts, a great variety of usernames and passwords were collected, namely mails, Git repository credentials, Windows domain passwords, sysadmin, browser stored credentials and passwords loaded on the random access memory of the targeted host.

6.2.6. Discussion upon Password Exploitation with LaZagne Project

With reference to the LaZagne Project password stealing attack, *ParentImage="C:/Users/ /Administrator/.../lazagne.exe"* was included to outline the execution of password spoofing LM attack with the use of the popular penetration framework *LaZagne Project*. The rule was enhanced with various extra *"ParentCommandLine"* features in an attempt to capture all the possible password stealing attempts from the targeted server's client *windows accounts*, *sysadmin*, *mails*, *databases etc.*, as presented in Listing 7.

```
<ProcessCreate onmatch="include">
   <ParentImage condition="is">C:\Users\Administrator\Downloads\lazagne.exe</ParentImage>
   <ParentCommandLine condition="is">lazagne.exe windows</ParentCommandLine>
   <ParentCommandLine condition="is"> lazagne.exe sysadmin</ParentCommandLine>
   <ParentCommandLine condition="is"> lazagne.exe mails</ParentCommandLine>
   <ParentCommandLine condition="is"> lazagne.exe project mails</ParentCommandLine>
   <ParentCommandLine condition="is"> lazagne.exe project databases</ParentCommandLine>
   <ParentCommandLine condition="is"> lazagne.exe project windows</ParentCommandLine>
   <ParentCommandLine condition="is"> lazagne.exe project all</ParentCommandLine>
</ProcessCreate>
```

Listing 7: lazagne.exe password spoofing

7. Python_Evtx_Analyzer

The analysis of large log files is always a demanding procedure for Incident response teams, struggling to meet with shortages on computational power to manipulate the millions of logs produced on a daily basis. Despite the variety in propriety and open-source Security Information and Event Management (SIEM) solutions for centralized logging and analyzing activities, most of them require multi-step procedures regarding log parsing, event analyzing, exception handling, and monitoring parameter initialization. That is, the literature lacks of a lightweight and easily configurable tool which can be used to automate the parsing and threat analysis of extended *.evtx* log sets. To cover this need, this section presents and evaluates a Python analyzing scripting tool dubbed *"Python_Evtx_Analyzer" (PeX)*, which caters for the analysis of voluminous Sysmon logs, and therefore contributes to the identification of LM events in a timely manner.

From a bird's eye view, *PeX* was developed to serve as a proof of concept for the proposed rule-based policy's efficiency discussed in section 6. Towards this goal, the optimization of Sysmon's logging activity is concentrated into filtering the less possible unwanted *noise* and being as targeted as possible to LM. Simply put, the essence of *PeX's* operation is to provide a portable and OS-independent command line (or IDE executable) tool, that helps EDR teams to analyze massive *.evtx* logs and successfully identify LM. It should be noted that *PeX's* events identification is based on LM-oriented features that were extracted from Sysmon's pre-configured rules in the enclosed config.xml file, as presented in Appendices A.1 and A.2. What makes *PeX* special is its ability to be fully customizable by Incident Response researching teams to analyse and identify any kind of logging activity captured by Sysmon, either normal or malicious. As a result, *PeX* can be used in the context of other researchers in this timely field as it is made publicly available as open source in [2].

In brief, the analyzer has dependencies on six Python libraries, namely *mmap*, *argparse*, *minidom*, *evtx.Evtx*, *evtx.Views* and *ElementTree)*, which should either be imported during the building of the IDE project or installed to the OS prior to the code execution with terminal. It allows the following functions:

- Memory mapping of massive Sysmon log files as inputs in .evtx form.
- Provision of basic arguments supporting a user-friendly command-line execution of the Python script's source code.
- Parsing and transformation of the logs included in the *.evtx* file into the easily manipulated and analyzed *.xml* format.
- Custom header filtering of the *.xml* created file. This is based on pre-selected Sysmon's EventID tags, presented in tree-based form on the command's line screen or stored externally to a *.txt* file.
- Manipulation of the tree-based *.xml* structure and filtering based on pre-configured features as enclosed in Sysmon's *config.xml*.
- Enumeration of the identified LM events, alert messages generation per attack denoting the number of the identified malicious events and making an assumption of the type of the potentially executed attack.

From an OS version's perspective, the analyzer can run on all mainstream platforms, including *Windows 10*, *MacOS Big Sur v11.6.5* and *Ubuntu v22.04*.

7.1. PeX operation

With all the prerequisite libraries imported, the main function's statements (*Function (def python_Evtx_Analyzer (-f , -i, -o) :)*) are executed revealing their potentials as presented in Algorithm 1. The latter describes the procedures of the input folder identification, *.evtx* files insertion, the output folder creation, and the buffering of the *.evtx* log headers in an algorithmic pseudocode format. Note that Algorithm 1 is implemented in conjunction with Algorithms 2 and 3 and run as a single Python script.

At first, the necessary arguments of input folder (*-f*), the ID of the *evtx* file (*-i*) and the optional output folder (*-o*) are imported and parsed via *Argparse* into (*"evtxAnalyzer"*) variable. Next, the existence of the *outputfolder* variable is checked if it is set to true and outputfolder is opened with the append (a+) permission. If set to false, the analyzer proceeds with terminal standard output, as presented in ll.6-7 of Algorithm 1.

Algorithm 1 PeX's arguments and folders initialization algorithm

```
Require: python $setup.py$ install
Require: pip install $mmap, argparse, minidom-ext, python-evtx$ libraries
Require: import $mmap, argparse, minidom\ (from\ xml.dom), evtx.Evtx.FileHeader, evtx.Views, xml.etree.ElementTree$
Require: $data, Sysmon$ files in $.evtx$ format
Require: Function def $python\_Evtx\_Analyzer(-f, -i, -o)$ :
    $evtxAnalyzer \leftarrow data[str][--iFolder, --evtxId, --outputFolder]$
    $arguments \leftarrow data[str][evtxAnalyzer.parse\_args()]$
 3: $outputFolder \leftarrow data[Boolean][False]$
    if $arguments.outputFolder \neq None$ then
        $outputFolder \leftarrow append("+a")permissions$
 6: else if $arguments.outputFolder == None$ then
        $return 0$
    end if
 9: for $arguments.ifolder \leftarrow read("r")permissions$ do
        $evtxBuffer \leftarrow read("r")permissions$
        $evtxBuffer \leftarrow data[Sysmon.evtxfiles][mmap.mmap(ifolder)]$
12:     $fileheader \leftarrow data[evtxBuffer][evtx.Evtx.FileHeader(evtxBuffer)]$
        $xmlHeaderOutput \leftarrow data[fileheader]$    ▷ The header of every xml file is stored to the xmlHeaderOutput variable.
        if $outputFolder == True$ then
15:         $outputFolder \leftarrow data[xmlHeaderOutput]$
        else if $outputFolder == False$ then
            $print(xmlHeaderOutput)$
18:     end if
    end for
```

The input folder contents are mapped via the *Memory Mapped file support (mmap)* library, stored into the *evtxBuffer*, to be set as input in the *evtx.Evtx FileHeader()* function. The buffered logs are enumerated based on their tag headers, to be finally stored in the *fileheader* variable. Recall that the *mmap* is a Python library which allows the manipulation through mapping of various input and output (I/O) file objects. The *fileheader* variable is manipulated as a collection of *xml* objects through *evtx.Views* library to eventually store the parsed headers to the *xmlToStr* variable. The collected *xml* entries of the *xmlToStr* object are parsed via the *Minidom* Python library, called *Minimal DOM*, based on *<EventID> tags* and stored in the *xmlToDoc* variable, as presented in Algorithm 2 (ll. 3).

Algorithm 2 PeX's evtx-to-xml transformation algorithm

```
    for $xmlToStr \leftarrow in[evtx.Views][fileheader]$ do
        $xmlToDoc \leftarrow [minidom.parseString][xmlToStr]$
3:      $eventsByID \leftarrow [getElementsByTagName][xmlToDoc]$
    end for
    if $arguments.id == "all"$ then
6:      if $outputFolder$ then
            $outputFolder \leftarrow data[xmlToDoc]$
        else if $arguments.outputFolder == None$ then
9:          $cmd(Terminal) \leftarrow data[xmlToDoc]$
        end if
    else if $eventsByID == arguments.id$ then
12:     if $outputFolder$ then
            $outputFolder \leftarrow data[xmlToDoc]$
        else if $arguments.outputFolder == None$ then
15:         $cmd(Terminal) \leftarrow data[xmlToDoc]$
        end if
    end if
```

The *ElementTree* module is combined with the "for tag in doc.findall("Name"):" loop and the desired pre-selected filtering features per attack, as presented in Algorithm 3 (ll. 3). The selected per attack case filters are iterated over a for loop against the provided as input .evtx file. The existence of each filter is counted and stored to a *"countVar"* variable. With the completion of the enumeration and filtering of the suspicious log file the analyzer prints two different messages to the user depending on the existence or not of suspicious network log traffic. In the case of attack's existence, the printed message is combined with the *"countVar"* variable to demonstrate the total number of identified packets. The results are summarized in a final report that is printed at the end of the analyzer's execution.

Algorithm 3 PeX's xml parsing and LM identification algorithm

```
    $counter == 0$
    $doc \leftarrow [ElementTree][xmlToStr]$
3:  for $tag \leftarrow [doc][findall("Name")]$ do
        if $tag \leftarrow [attrib][SysmonEventIDField] == "PreselectedValue\_01"$ or $tag \leftarrow [attrib][SysmonEventIDField] == "PreselectedValue\_02"$ then
            $doc \leftarrow [remove][tag]$
6:      end if
        $print \leftarrow [ElementTree][doc]$
    end for
9:  for $countVar \leftarrow [xmlToStr][count("PreselectedValue")]$ do
        if $countVar >= 1$ then
            $counter++$
12:         $print \leftarrow [counter][Windows\_terminal\_Alert\_Message.]$
        end if
        $print \leftarrow [ElementTree][doc]$
15: end for
    $print \leftarrow [Analyzer\_Final\_Report][doc]$
    $evtxBuffer \leftarrow [close()]$
18: $return$
```

7.2. Dataset

PeX was evaluated over a 10-days dataset, regarding the analyzer's detection and alerting rates. Those are processed as attacks-related fractions based on the detected rule-based logs, along with the False Positives (FP) and False Negatives (FN) alert messages it generates. The collected data contain normal and malicious logs generated from the continuous interaction with the constituent systems of the SOHO network of Figure 2. Specifically, the nine LM attacks of Section 6 were re-executed multiple times over the 10-days period and mixed with legitimate user activity upon the targeted SOHO network. The legitimate traffic includes the captured logs of the first day, consisted basically from user logins and system logouts, web browser usage and Internet surfing, file sharing among the various stations of the network, email traffic and various social media accounts logins and user interactions with each of the six client stations included in Table 2. For reasons of reproducibility, but also for advancing research efforts in this area, the resulting dataset is publicly offered at [2].

Regarding the generated malicious traffic, the three variants of the *Exploitation of Remote Services* LM methodology were executed continuously for the first three days in a row simultaneously with the normal utilization network traffic. The next four days were devoted to the execution of the *Credential Exploitation Hashing Attacks*. Finally, the last two days of the 10-days testing period were concentrated on a mixed repetition of the aforesaid attacks, leading in the creation of an $\approx 870,000$ logs dataset of Sysmon *.evtx* files. It should be noted that each day represents 3 to 6 hours of continuous capturing of network logs activity and not a 24 hours whole.

7.3. Evaluation

As a proof of concept, *Pex* was implemented with Python 3 on a VM Linux machine with 16 GB of RAM and a quad-core processor. As presented in Table 6, four different subsets of the pre-collected logs were extracted, namely, *Normal* (or $\approx 80,000$ EventIDs), *NormalVsMalicious01* (or $\approx 290,000$ EventIDs), *NormalVsMalicious02* (or $\approx 415,000$ EventIDs) and *FullSet* (or $\approx 870,000$ EventIDs). Table 7 summarizes the total number of tests that were conducted with *PeX* per attack and subset, including the number of logs and TP/FP rates. All the experiments were performed using the extended rule-based filtering mode presented in Appendices A.1 and A.2.

Indicatively, *Pex* took an average analyzing time of ≈15 min and 14 sec of CPU time regarding the first three subsets of Table 6. As it concerns the *FullSet*, the analyzer took an average of 16 min and 22 sec. This is translated to an increase by ≈ 7.13% vis-à-vis the average processing and analysis time that was that was perceived for the first three subsets of Table 6.

Regarding the conducted experiments, *PeX* successfully achieved an average detection rate above 90%, as it concerns the summed percentage of TP and FN identified incidents per subset. On the other hand, during each *.evtx* subset analysis, *PeX* generated a mean rate of ≈ 10% *FP* and ≈ 1.5% *FN* misidentification incidents respectively.

Table 6: Subsets examined with *PeX*, including the total number of *.evtx* logs per subset.

Tested Subsets	**Sysmon EventIDs (logs)**
Normal	$\approx 80,000$
NormalVsMalicious01	$\approx 290,000$
NormalVsMalicious02	$\approx 415,000$
FullSet	$\approx 870,000$

PeX evaluation started with the *Normal* subset and was examined under the proposed rule-based policy of Appendices A.1 and A.2. First, the logs were tested against the proposed rule-based filtering features, without the implementation of the identified Mimikatz's *".dll"* rules given in Table 5. As depicted in Table 7, *PeX* successfully identified 70,641 EventID logs (or else 88% TP rate) related to Normal network traffic, plus another $\approx 2,000$ logs revealing a 2.5% of TN results. Overall, it achieved a score of 90.5% regarding the identification of *Normal* network traffic, as the sum of TP and TN rates. On the other hand, the analyzer misidentified 7,224 logs as malicious revealing a tendency of 9% on FP events regarding the *"Normal"* subset. The majority of FP events are related to the fictitious presence of Mimikatz's *ProcessCreate EventIDS*, specifically the logs with *TargetImages: "C:/Windows/System32/lsass.exe, services.exe, reg.exe, svchost.exe"* and *GrandedAccess: "0x1010, 0x1410"*.

On the positive side, despite the initial concerns that a FP rate of ≈ 10% may arise, this figure falls under 1.5% (1.35% precisely) when combined with the *".dll"* related rules of Table 5. The aforesaid percentage, if examined in conjunction with the 1.5% of the fourth column's FN events, it is statistically acceptable for the operational nature of a parsing and analysis tool. The reader should keep in mind that the FN events presented in Table 7 are related to Sysmon's rules that were generated and implemented in *config.xml* file for ignoring any unwanted noise within the event filtering procedure.

Regarding the *NormalVsMalicious01* subset, which comprises logs related to the experiments of Subsection 6.1, *PeX* successfully identified 250,666 (or 85.4%) TP logs as potentially malicious or normal together with 15,556 (or 5.3%) more TN EventIDS. The same promising rates regarding TP and FP logs, were also identified during the evaluation of the *NormalVsMalicious02* subset of password hashing Exploitation attacks of subsection 6.2. Precisely, 83.5% (or 347,384 logs) of the collected logs was successfully identified, while another 3.3% (or 13,728 logs) was also flagged correctly as TN.

With reference to the penultimate column of Table 7, FP presented an average of 9.75% on both the *NormaVsMalicious* subsets. In more detail, *PeX* recognized incorrectly 23,481 (or 8%) and 47,843 (or 11.5%) logs per *NormaVsMalicious* subset, respectively. It is noteworthy that most of the misclassified logs of both subsets were related to Mimikatz extracted log files. For this reason, these records were re-evaluated under the implementation of the *.dll ParentImage* signatures of

Table 7: Total number of tests performed with *PeX* per attack and subset, including FP and TP percentages.

Attack Vector	Subset	No. of logs	TP (%)	TN (%)	FP (%)	FN (%)
Normal Network Traffic	*Normal*	80,274	88.0%	2.5%	9.0%	1.5%
Remote Services Exploitation + Normal Net Traffic	*NormalVsMalicious01*	293,520	85.4%	5.3%	8.0%	1.3%
Hashing Pass Exploitation + Normal Net Traffic	*NormalVsMalicious02*	416,029	83.5%	3.3%	11.5%	1.7%
Total Attacking Vector + Normal Net Traffic	*FullSet*	870,119	84.6%	2.4%	11.6%	1.4%

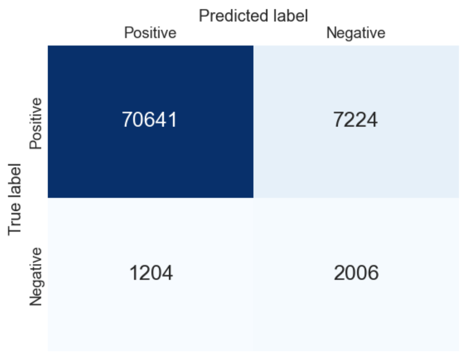
Table 5, reducing erroneous FP rates in 1.9% and 2.3% for each subset, as presented in Table 8, respectively. Regarding the identified FNs per subset, the records of Table 8 do not exceed 1.6%, a fact that within the context of this work it is deemed acceptable.

Table 8: Total number of tests performed with *PeX* per attack and subset, including FP and TP percentages. The evaluation of the depicted logs was executed under the extended rule-based policy, including the *".dll"* analysis of Table 5.

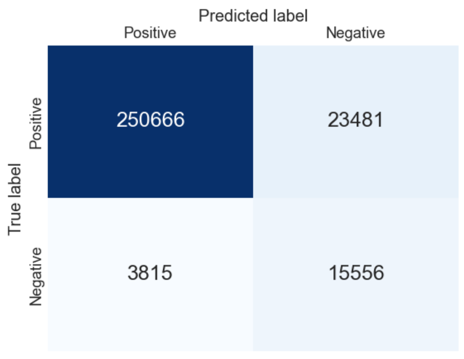
Subset	No. of logs	TP (%)	TN (%)	FP (%)	FN (%)
Normal	80,274	89.63%	7.60%	1.35%	1.42%
NormalVsMalicious01	293,520	88.65%	8.20%	1.90%	1.25%
NormalVsMalicious02	416,029	88.25%	7.85%	2.30%	1.60%
FullSet	870,119	88.55%	7.90%	2.20%	1.35%

No less important, *PeX* was tested against the extended *FullSet* dataset, which comprises the three pre-selected log records of Table 7 plus the events of the last two days of the 10-days period testing. The final evaluation of the analyzer was conducted under the eventually configured rule-based policy, including the *".dll"* filtering parameters. Table 7 presents *PeX* scoring without the inclusion of the *.dll* EventID's fields, while Table 8 contains the re-evaluated log values under the enhanced rule-based policy.

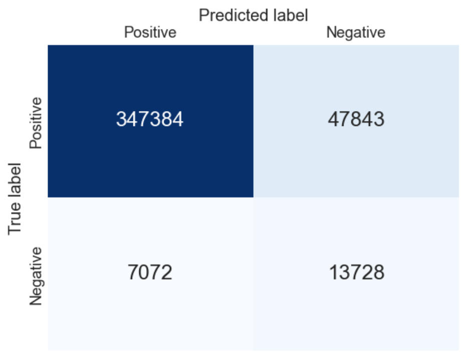
Last but not least, the confusion matrices presented in Figures 5 and 6 demonstrate in a log-oriented form, the evaluation percentages of Tables 7 and 8, namely TP, TN, FP, FN rates, for *PeX's* execution per technique and subset.



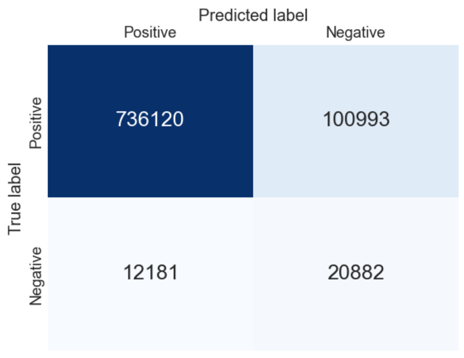
(a) Normal Subset



(b) NormalVsMalicious01 Subset

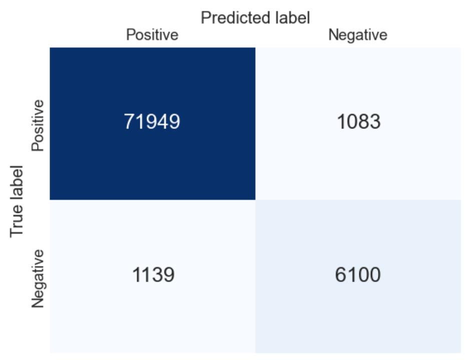


(c) NormalVsMalicious02 Subset

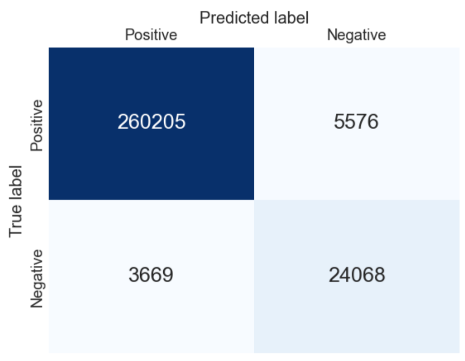


(d) FullSet Subset

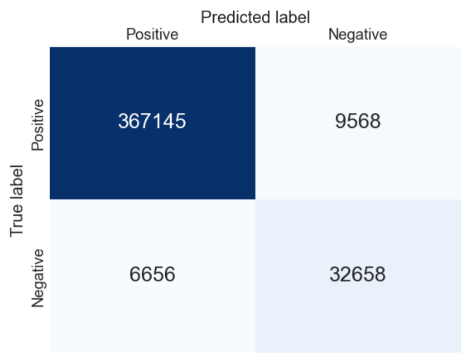
Figure 5. Confusion matrices for the evaluation percentages presented in Table 7, namely TP, TN, FP, FN rates, for *PeX's* execution per technique and subset.



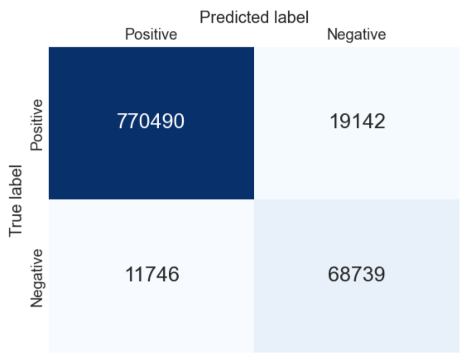
(a) Normal Subset



(b) NormalVsMalicious01 Subset



(c) NormalVsMalicious02 Subset



(d) FullSet Subset

Figure 6. Confusion matrices for the evaluation percentages presented in Table 8, namely TP, TN, FP, FN rates, for *PeX's* execution per technique and subset.

8. Conclusions

In recent years, many organizations have suffered damages as a result of targeted LM. As it becomes apparent, the tasks of incidence response teams to further address such damages acquire special weight. Many penetration testing tools leave no evidence regarding their execution when filtered with Sysmon's default settings, resulting in many event infiltration issues being unsolved. In this context, the current study aspires to set the ground for a fresh EDR rule-based policy regarding Sysmon identification and alerting of LM incidents. Besides that, a publicly available Python tool, namely *PeX* is implemented, giving greater insight in terms of understanding the mechanisms that fall under LM methods.

As an extension to the so far conducted work in this topic, the ultimate goal of the current work is the collection and investigation of evidential logs related to LM attacks. In short, the effort is concentrated on the creation of custom filtering Sysmon rules, along with the initialization of Sysmon's config.xml file specifically oriented towards the alerting of LM. The conducted analysis suggests two different sets of Sysmon rules, as presented in Appendices A.1 and A.2, one for each of the Exploitation of Remote Services and Credential Exploitation attack's experiments, respectively. The adoption of the aforementioned rules comes as a direct result of the thorough network traffic and pattern study, as derived from the execution of each attack.

With reference to the conducted experiments, the proposed rule-based approach was incorporated with Sysmon in the form of the *config.xml* file, presenting an identification TP rate above $\approx 95\%$. Finally yet importantly, the proposed EDR policy, including the *.dll* analysis of Table 5, was imported and evaluated through *PeX* against four subsets, revealing a tendency above $\approx 96\%$ in terms of the TP and TN metrics. The combination of the proposed EDR policy with machine learning techniques, may reveal its importance to act as the basis for the creation of a LM-oriented IDS solution. Nevertheless, a thorough investigation of this potential is well beyond the scope of this paper and is left for future work.

Abbreviations

The following abbreviations are used in this manuscript:

5Ws	Who – When – Where – What - Why
Argparse	Argument Parser Python Library
API	Application Programming Interface
ARP	Address Resolution Protocol
CMD	Command Line
CME	CrackMapExec
CSIRT	Computer Security Incident Response Team
CSV	Comma-Separated Values
DLL	Dynamic Link Library
DOM	Document Object Model Interface
Domain SID	Domain Security Identifier
EDR System	Endpoint Detection and Response System
Event ID	Event Identification
evtx	Windows XML EventLog
FN	False Negative
FP	False Positive
GUID	Global Unique Identifier
I/O	Input / Output
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IP	Internet Protocol
KDC	Kerberos Key Distribution Center
klist	Kerberos List
LSASS	Local Security Authority Subsystem Service
LTS	Long Time Support
MAC Address	Media Access Control Address
MMAP	Memory Mapped File Support Python Library
MSTSC	Microsoft Terminal Services Client
NMAP	Network Mapper
NTLM	Network Lan Manager
PeX	Python_Evtx_Analyzer
PtH	Pass the Hash
PtT	Pass the Ticket
RAM	Random Access Memory
RDP	Remote Desktop Protocol
SIEM	Security Information and Event Management
SMB	Service Message Block
SOHO	Small Office Home Office
STA	Station
Sysmon	System Monitoring
TN	True Negative
TP	True Positive
UAC	User Account Control
USB	Universal Serial Bus
VM	Virtual Machine
WIDS	Wireless Intrusion Detection Systems
Wi-Fi	Wireless Fidelity
xml	Extensible Markup Language

References

1. MITRE. Lateral Movement - The adversary is trying to move through your environment., July 2019.
2. Smiliotopoulos, C.; Barbatsalou, K.; Kambourakis, G. Python_Evtx_Analyzer (PeX - v1). https://github.com/ChristosSmiliotopoulos/Python_Evtx_Analyzer.git, accessed on 2022.
3. Coordination, J. Detecting lateral movement through tracking event logs, June 2017.

4. Russinovich, M.; Garnier, T. Sysmon v13. 22. Retrieved June **2021**, 28, 2021.
5. Coordination, J. Detecting Lateral Movement through Tracking Event Logs (Version 2), December 2017.
6. Viasat. KaSat - Network cyber attack overview, 2022.
7. Mavroeidis, V.; Jøsang, A. Data-driven threat hunting using sysmon. Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, 2018, pp. 82–88.
8. Mavroeidis, V.; Bromander, S. Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence. 2017 European Intelligence and Security Informatics Conference (EISIC), 2017, pp. 91–98. doi:10.1109/EISIC.2017.20.
9. Berady, A.; Jaume, M.; Tong, V.V.T.; Guette, G. From TTP to IoC: Advanced Persistent Graphs for Threat Hunting. IEEE Transactions on Network and Service Management **2021**, 18, 1321–1333. doi:10.1109/TNSM.2021.3056999.
10. Matsuda, W.; Fujimoto, M.; Mitsunaga, T. Real-Time Detection System Against Malicious Tools by Monitoring DLL on Client Computers. 2019 IEEE Conference on Application, Information and Network Security (AINS), 2019, pp. 36–41. doi:10.1109/AINS47559.2019.8968697.
11. Juwono, J.T.; Lim, C.; Erwin, A. A comparative study of behavior analysis sandboxes in malware detection. International Conference on New Media (CONMEDIA), 2015, p. 73.
12. Narouei, M.; Ahmadi, M.; Giacinto, G.; Takabi, H.; Sami, A. DLLMiner: structural mining for malware detection. Security and Communication Networks **2015**, 8, 3311–3322.
13. Rajesh, P.; Ismail. Ismail. B, M.; Alam, M.; Tahernezhadi, M. Network Forensics Investigation in Virtual Data Centers Using ELK. 2021 International Symposium on Electrical, Electronics and Information Engineering, 2021, pp. 175–179.
14. Jain, U.; others. Lateral movement detection using ELK stack. PhD thesis, University of Houston, 2018.
15. El-Hadidi, M.G.; Azer, M.A. Detecting Mimikatz in Lateral Movements Using Mutex. 2020 15th International Conference on Computer Engineering and Systems (ICCES), 2020, pp. 1–6. doi:10.1109/ICCES51560.2020.9334643.
16. Ki, Y.; Kim, E.; Kim, H.K. A Novel Approach to Detect Malware Based on API Call Sequence Analysis. Int. J. Distributed Sens. Networks **2015**.
17. Ho, G.; Dhiman, M.; Akhawe, D.; Paxson, V.; Savage, S.; Voelker, G.M.; Wagner, D. Hopper: Modeling and Detecting Lateral Movement. 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 2021, pp. 3093–3110.
18. Bhasin, H.P.S.; Ramsdell, E.; Alva, A.; Sreedhar, R.; Bhadkamkar, M. Data center application security: Lateral movement detection of malware using behavioral models. SMU Data Science Review **2018**, 1, 10.
19. Coffey, K.; Smith, R.; Maglaras, L.; Janicke, H. Vulnerability analysis of network scanning on SCADA systems. Security and Communication Networks **2018**, 2018.
20. Mulder, J.; Stingley, M. Mimikatz overview, defenses and detection. SANS Institute, retrieved from the Internet, https://www.sans. org/reading-room/whitepapers/detection/mimikatz-overview-defenses-detection-36780 Feb **2016**, 18.
21. Ussath, M.; Jaeger, D.; Cheng, F.; Meinel, C. Advanced persistent threats: Behind the scenes. 2016 Annual Conference on Information Science and Systems (CISS). IEEE, 2016, pp. 181–186.
22. Kazanciyan, R.; Hastings, M. Investigating powershell attacks. Black Hat **2014**, p. 25.
23. F, S. Revisiting Remote Desktop Lateral Movement. Medium, retrieved from the Internet, https://posts.specterops.io/ **2022**.
24. logPoint. MITRE ATT&CK Analytics — Alert Rules latest documentation. Docs.logpoint.com, retrieved from the Internet, https://docs.logpoint.com/docs/alert-rules/en/latest/MITRE.html **2022**.
25. Goet, M. Protect yourself against #BlueKeep using Azure Sentinel and Defender ATP. Medium, retrieved from the Internet, https://medium.com/@maarten.goet/protect-yourself-against- bluekeep-using-azure-sentinel-and-defender-atp-d308f566d5cf **2022**.
26. SandBoxCloud, J. Windows Analysis Report. SandBoxCloud, retrieved from the Internet, https://www.joesandbox.com/analysis **2022**.
27. Sandbox, F. Free Automated Malware Analysis Service, 'RemoteDesktopManagerFree.exe'. Sandbox, retrieved from the Internet, https://www.hybrid-analysis.com/sample/ b26ede46a0be62f361b4a28d2e67fa2e2f35c9bbc99 5ae84a2c0c7f4141f65b0 ?environmentId=100 **2022**.
28. Sorensen, S. Remote Desktop Manager Free. LO4D.com, retrieved from the Internet, https://remote-desktop-manager-free.en. lo4d.com/windows **2022**.
29. Bezverkhyi, A. Proactive detection content: CVE-2019-0708 vs ATT&CK, Sigma, Elastic and ArcSight - SOC Prime. SOC Prime, retrieved from the Internet, https://socprime.com/blog/proactive-detection-content-cve-2019-0708-vs-attck-sigma-elastic-and -arcsight/ **2022**.
30. s0i37. Lateral movement guide: Remote code execution in Windows – HackMag. Hackmag.com, retrieved from the Internet, https://hackmag.com/security/lateral-guide/ **2022**.

Appendix A

Appendix A.1.

Table 10: Summary of the most important rule-based features included in subsection 6.1. The imported superscripts are described as follows: Number 1 of the first column denotes Microsoft's vulnerability with code MS-17-010 "EternalChampion SMB Remote Command Execution", Number 2 indicates Impacket Python scripts exploitation, Number 3 is related with "Escalation to SYSTEM privilege with Metasploit smb_login exploitation", Number 4 presents UAC bypass - privilege escalation with smbexec.py, Number 5 is the exploitation of the EternalBlue vulnerability on SMB file-sharing services, and Number 6 demonstrates BlueKeep RDP misconfiguration most related features.

Summary of the most important rule-based features included in Sub-section 6.1

Tool	Tool Attack Utilization	COM Port	Sysmon - Event Log
[1], [2]PsExec	Remote command execution on client and servers within a domain.	135/tcp, 445/tcp, or a random high port	**EventID 1,5** (ProcessCreate, ProcessTerminate), **path:** "%SystemRoot%/PSEXESVC.exe", **Images:** "C:/Windows/PSEXESVC.exe", "C:/Windows/services.exe", "C:/Windows/wmiprsv.exe" User: "SYSTEM", **Additional Info:** "date, UtcTime the Image: "C:/Windows/PSEXESVC.exe" was executed".
[1], [2]WinRM	Target enumeration - investigation on remote host via command execution.	5985/tcp (HTTP) or 5986/tcp (HTTPS)	**EventID 1,5** (ProcessCreate, ProcessTerminate), **Images:** "C:/Windows/System32/cscript.exe", "C:/Windows/System32/svchost.exe", "C:/Windows/System32/wsmprovhost.exe", "C:/Windows/System32/mmc.exe", **Additional Info:** ProcessID, UtcTime, CommandLine, User.
[2], [4]smbexec	Execution of applications that are normally controlled by the bypassed User Account Control (UAC) as a user with administrator privileges.	-	**EventID 1,5** (ProcessCreate, ProcessTerminate), **Images:** "C:/Windows/System32/smbexec.py", "C:/Windows/System32/sdbinst.py" User: "SYSTEM", **Additional Info:** "Process Start/End Time and Date (UTC), Process CommandLine, ProcessID, SDB File Used".
[1], [2]smb_login (Metasploit)	Privilege Escalation within host or domain.	445/tcp	**EventID 1,5** (ProcessCreate, ProcessTerminate), **path:** "%SystemRoot%/PSEXESVC.exe", **Images:** "C:/Windows/PSEXESVC.exe", "C:/Windows/System32/cmd.exe", "C:/Windows/System32/powershell.exe", "C:/Windows/System32/net.exe", "C:/Windows/System32/whoami.exe" **Additional Info:** Administrative privileges "//*/IPC$", "//*/admin$", "//*/c$", Images - CommandLine - CurrentDirectory - Hashes - ParentImage - ParentCommandLine related to net.exe and whoami.exe.

Continued on next page

doi:10.20944/preprints202207.0176.v1

Table 10 – continued from previous page

Summary of the most important rule-based features included in subsection 6.1			
Tool	**Tool Attack Utilization**	**COM Port**	**Sysmon - Event Log**
[5]Wannacry malware	Malware infection to leverage MS17-010 EternalBlue vulnerability on SMB file-sharing services.	139/tcp, 445/tcp, 137-138/udp	**EventID 1,2,5** (ProcessCreate, ProcessChange, ProcessTerminate), **path:** "%SystemRoot%/Wcry.exe", "%SystemRoot%/taskce.exe" **Images:** "C:/Windows/Users/Desktop/Wcry.exe", "C:/Windows/Users/Desktop/tasksche.exe", "C:/Windows/Users/Desktop/r.wnry.exe", "C:/Windows/Users/Desktop/s.wnry.exe", "C:/Windows/Users/Desktop/t.wnry.exe", "C:/Windows/Users/Desktop/u.wnry.exe", "C:/Windows/Users/Desktop/taskse.exe", "C:/Windows/Users/Desktop/taskdl.exe", "C:/.../attrib.exe", "C:/.../icacls.exe" **Additional Info: ProcessID:** 3024, Files with extension ".WNCRYT" or "C://Users//...//Desktop//Eula.txt.WNCRYT" objects which denote ransomware's expansion, **CommandLine:** "attrib +h", "icacls . /grant Everyone:F /T /C /Q".
[6]BlueKeep (CVE-2019-0708)	Worm-like (Wannacry similiral) cybersecurity vulnerability of Windows Remote Desktop Protocol (RDP) and Services (RDS) which allows the remote execution of arbitrary malicious code.	3389/tcp	**EventID 1,2,5** (ProcessCreate, ProcessChange, ProcessTerminate) as in [1−5], **EventID 3** (NetworkConnection), **path:** "%SystemRoot%/", **Images:** "C:/Windows/System32/rdpv.exe", "C:/Windows/System32/mstsc.exe", "C:/Windows/System32/RTSApp.exe", "C:/Windows/System32/ws_TunnelService.exe.exe", "C:/Windows/System32/RemoteDesktopManagerFree.exe", "C:/Windows/System32/RemoteDesktopManager.exe", "C:/Windows/System32/RemoteDesktopManager64.exe", "C:/Windows/System32/mRemote.exe", "C:/Windows/System32/Terminals.exe", "C:/Windows/System32/spiceworks-finder.exe", "C:/Windows/System32/thor.exe", "C:/Windows/System32/thor64.exe", "C:/Windows/System32/reg.exe", "C:/Windows/System32/chrome.exe", **Sysmon's Additional Info:** Administrative privileges "//*/IPC$", "//*/admin$", "//*/c$", Images - CommandLine - CurrentDirectory - Hashes - ParentImage - ParentCommandLine - ProcessID - UtcTime, User related to the identified **Images**.

This appendix demonstrates the most important rule-based features included in subsection 6.1. The imported superscripts are described as follows.

Appendix A.2.

Table 11: Summary of the most important rule-based features included in Sub-section 6.2. The imported superscripts are described as follows: Number 1 presents the most significant features of the PtH attack, Number 2 , Number 3-5 are dedicated to the description of the Pass-the-Ticket (PtT), Golden and Silver Tickets attacks, Number 5 presents LaZagne Project related features regarding Windows passwords compromising, and Number 6 is related to the Privilege Escalation and enumeration procedure.

Summary of the most important rule-based features included in Sub-section 6.2

Tool	**Tool Attack Utilization**	**COM Port**	**Sysmon - Event Log**
[1] PtH (PowerShell Invoke-Mimikatz)	Credential theft and Lateral Movement technique in which the adversary leverages Windows NTLM hash without cracking it to authenticate as legit.	445/tcp	**EventID 1,2,5** (ProcessCreate, ProcessChange, ProcessTerminate) as in $^{1-6}$ of Table 10, **EventID 3** (NetworkConnection), **EventID 10** (ProcessAccess), **EventID 11** (FileCreate), **EventID 22** (DNSEvent (DNS query)), **EventID 13** (RegistryEvent (Value Set)), **path:** "%SystemRoot%/", "%SystemRoot%/PSEXESVC.exe", **ParentImage:** "C:/Users/Administrator/Desktop/mimikatz_trunk /x64/mimikatz.exe", "C:/Windows/system32/sppsvc.exe" **TargetImage:** "C:/Windows/System32/lsass.exe", "C:/Windows/System32/lsass.exe", "C:/Windows/System32/reg.exe", "C:/Windows/System32/svchost.exe", "C:/Windows/PSEXESVC.exe", "C:/Windows/services.exe", "C:/Windows/wmiprsv.exe" **GrantedAccess:** "0x1010", "0x1410", "x1438", "x1438a", "0x1FFFFF", "0x143a", **CommandLine:** "sekurlsa", "mimikatz", "reg SAVE", "dumpcr", **Sysmon's Additional Info:** Administrative privileges "//*/IPC$", "//*/admin$", "//*/c$", Images - CommandLine - CurrentDirectory - Hashes - ParentImage - ParentCommandLine - ProcessID - UtcTime, User related to the identified **Images**, TargetFilename: "C:/Users/Administrator/Desktop/.../filename.kirbi", DNS QueryName: "sysmon_set (or the desired name of the targeted domain)"
[2] PtT,[3] Golden Ticket,[4] Silver Ticket	Capturing the Domain Administrator Privilege and Account Credentials. Leverages an unauthorized Kerberos ticket that is valid for an arbitrary period and grants access without additional authentication.	445/tcp	**EventID 1,2,5** (ProcessCreate, ProcessChange, ProcessTerminate), **EventID 3** (NetworkConnection), **EventID 10** (ProcessAccess), **EventID 11** (FileCreate), **EventID 22** (DNSEvent (DNS query)), **EventID 13** (RegistryEvent (Value Set)), **path:** "%SystemRoot%/","%SystemRoot%/PSEXESVC.exe", as in $^{1-7}$ of Table 10, **ParentImage:** as in $^{1-7}$, plus, C:/Windows/System32/klist.exe, C:/Windows/mimikatz.exe, "C:/Windows/PSEXESVC.exe", **CommandLine:** "sekurlsa", "mimikatz", "reg SAVE", "dumpcr", "Get-Keystrokes", "Get-TimedScreenshot", "Get-VaultCredential", "Invoke-CredentialInjection", "Invoke-CredentialInjection"

Continued on next page

Table 11 – continued from previous page

Summary of the most important rule-based features included in Sub-section 6.2

Tool	**Tool Attack Utilization**	**COM Port**	**Sysmon - Event Log**
[5] LaZagne	Open source application used for passwords exploitation which are stored on a local targeted host.	3389/tcp	**EventID 1,2,5** (ProcessCreate, ProcessChange, ProcessTerminate), **EventID 10** (ProcessAccess), **path:** "%SystemRoot%/","%SystemRoot%/PSEXESVC.exe", **ParentImage:** "C:/Windows/System32/lazagne.exe", **CommandLine:** "lazagne.exe windows", "... sysadmin, mails, project mails, project databases, project windows, project all"
[6] Privilege Escalation	The basis of each Lateral Movement attack. Target acquisition, weak points enumeration, access gaining within the security perimeter of a domain, gradual escalation of privileges and extension of the Lateral movement of the adversary.	80, 53, 25, 110, 143, 139, 445, 3389, 6000/TCP	**EventID 1,2,5** (ProcessCreate, ProcessChange, ProcessTerminate), **CommandLine:** "Get-Keystrokes", "Get-TimedScreenshot", "Get-VaultCredential", "Invoke-CredentialInjection", "mimikatz"