advances in Technology

Adv. Technol. **2022**, 2(3), 291-321

Full Paper

A Novel Method for Moving Laterally and Discovering Malicious Lateral Movements in Windows Operating Systems: A Case Study

Kyle T. Rozendaal[a] and Akalanka B. Mailewa[b,*]

[a]Department of Information Assurance, College of Business, St. Cloud State University, Minnesota 56301, USA

[b]Department of Computer Science & Information Technology, St. Cloud State University, St. Cloud, Minnesota 56301, USA

E-mail correspondence: amailewa@stcloudstate.edu (A. B. Mailewa)

Received: 15 March 2022; Revised:14 June 2022; Accepted: 18 June 2022; Published: 25 August 2022

Abstract

Lateral movement is a pervasive threat because modern networked systems that provide access to multiple users are far more efficient than their non-networked counterparts. It is a well-known attack methodology with extensive research conducted investigating the prevention of lateral movement in enterprise systems. However, attackers use increasingly sophisticated methods to move laterally that bypass typical detection systems. This research comprehensively reviews the problems in lateral movement detection and outlines common defenses to protect modern systems from lateral movement attacks. A literature review outlines techniques for automatic detection of malicious lateral movement, explaining common attack methods utilized by advanced persistent threats and components built into the Windows operating system that can assist with discovering malicious lateral movement. Finally, a novel approach for moving laterally designed by other security researchers is reviewed and studied, an original process for detecting this method of lateral movement is proposed, and the application of the detection methodology is also expanded.

Keywords: Lateral-movements, Cyber-attacks, Confidentiality, Security, Phishing, Threats, ShadowMove

---

Introduction

Lateral movement is a technique outlined in the MITRE ATT&CK framework and is a major problem in enterprise networks during cyber-attacks [1]. Lateral movement takes place after attackers gain a foothold in a network. Attackers use a combination of built-in programs, malware, remote procedure calls, and user-agent manipulation to move between workstations and servers closer to the target system containing the data they wish to manipulate for personal benefit. Many researchers focus on earlier phases of the attack chain because once lateral movement has begun, the attackers have already breached the perimeter, making it extremely difficult to contain the damage. Phishing is vital in explaining why detecting and preventing lateral movement is essential. According to phishlabs.com and the 2019 DBIR by Verizon, Phishing attacks were a crucial component in 32% of all successful data breaches [2][3]. Typically, when a phishing attack is successful, there is a loss in the confidentiality of the user account and password, or malware is downloaded into the network giving attackers remote access to the Windows environment. A successful phish can bypass multiple steps on the ATT&CK framework [1] if a high-level administrator reveals their username and password information. Likewise, even compromising the integrity of a standard user could bypass a

https://doi.org/10.31357/ait.v2i3.5584

few steps on the ATT&CK framework and allow access to attackers to systems allowing them to build persistence within the network from which to launch privilege escalation attacks against internal vulnerabilities. Since users are one of the weakest links in network security and a successful phish can bypass numerous defensive measures, research into detecting and preventing lateral movement is important in cyber-security and threat intelligence. Lateral movement has been challenging to detect in the past since the ability to move laterally between systems is a crucial component in networked Windows environments. Many technical users need access to multiple systems throughout the day; protocols like remote desktop protocol (RDP) and Secure Shell (SSH) are important business tools to help companies achieve their goals. However, the same tools and methods users need also allow attackers to move between computers and get into critical systems from which they can exfiltrate their target data. A key example of how lateral movement is an issue for businesses occurred in 2013 when Target experienced one of the largest data breaches of the decade. Target contracted an HVAC company to run HVAC units that could be remote controlled to save on heating and cooling costs during off-hours. These HVAC systems had remote capabilities so managers could adjust store temperatures and control costs. The attacker’s first entry into Target’s computer systems was through weak security protocols on these remote HVAC systems. Attackers then used lateral movement techniques to move into the point-of-sale terminal systems and exfiltrate nearly 40 million customers’ credit card information, costing Target nearly $300 million [4][5].

The Nature & Significance of the Problem

Lateral movement is a pervasive threat during cyber-attacks and often closely resembles legitimate traffic [6]. Attackers take advantage of the difficult nature of detection to move laterally through systems unnoticed. Modern detection currently relies on tried-and-true methods that detect standard lateral movement techniques, but new attack vectors are being developed, and modern systems need a refresh to keep up with novel attack patterns.

Lateral Movement is a major problem in security breaches

As previously stated, lateral movement is executed during almost every cyber-attack. In a large production environment, it is improbable that a server hosting sensitive data like credit card information, medical records, or banking information would be open to the public internet. Therefore, if attackers want access to sensitive information, the initial point-of-compromise will be any public-facing machine they can find. Once attackers gain access to a device, they will use the tools at their disposal to move from system to system until they get to the data, they are attempting to exfiltrate. Since lateral movement occurs in every large data breach, it is a significant problem to study.

Lateral Movement is Preventable

Lateral movement is entirely preventable. By locking down firewall rules and disabling protocols, lateral movement attacks would be impossible to execute by creating a series of systems that have no interconnectivity. However, preventing attackers from moving laterally also prevents legitimate users from moving laterally. Networked computers, systems, and a series of machines working in tandem with shared access make a business more profitable and efficient. It is understood that a computer connected to the internet is vastly superior in its capabilities to a computer without an internet connection. Likewise, a non-networked workstation in business environments would be significantly less powerful and productive than a workstation connected to the business network with access to all the business data. Due to the complexity of business systems in a networked world, shutting down methods to move laterally is impossible. Businesses rely on networked machines and servers to carry out business goals. MITRE supplies a list of controls that can be used to prevent some of the main types of lateral movement techniques. When implementing controls, there is a balance between system usability and security. Administrators must decide where this balance is between secure and useful. Since networked systems are the key to many business practices, differentiating between normal user movement through a networked system and malicious movement in an enterprise network is extremely important in supporting the security of data systems. Since remote access to networked computers is vital to business function, detection and prevention of unauthorized access are as important as the business functions.

Lateral Movement is difficult to detect

Attackers use numerous methods for moving laterally through networks [7]. Many of these methods require the attackers to capture password hashes and user credentials to move onto the next system. When attackers move this way, it simply appears that a named user is moving from one system to the next. This, in turn, means that attackers typically masquerade as many different users during a campaign, and all the lateral movement is masked to appear as though authenticated and authorized users are conducting legitimate business. However, some indicators of lateral movement show that illegitimate lateral movement is taking place rather than authorized remote work. Researchers have zeroed in on some of these digital artifacts created during a malicious lateral movement campaign and designed their methodologies for detecting these slight differences.

Objective of the Study

The study aims to supply a broad overview of lateral movement techniques and defenses. First, online sources and libraries like MITRE have troves of information concerning the practical application of lateral movement prevention and the methods protected against it. Second, a literature review covers proposed methods for improved lateral movement detection in enterprise systems. Finally, this study will demonstrate a novel method presented in one of the research papers, explain how it functions to bypass detection, and create a detection method to detect this stealthy lateral movement. This feat has not yet been completed.

Research Questions

a. Traffic Differentiation

With standard attack methodologies, how can an analyst differentiate between legitimate and non-legitimate traffic when legitimate traffic looks just like malicious traffic?

b. Network Complexity

With the increase in network complexity and the ever-increasing scope of networks, what defense measures have been proposed to assist analysts with detecting and preventing malicious traffic?

c. Novel Techniques

With the increasing complexity in detection evasion employed by attackers, is it possible to detect novel lateral movement techniques that have not yet been detected?

Limitations of the Study

Since lateral movement is a well-known and well-documented attack methodology, there is a significant amount of documentation outlining the methods and defenses found in research papers, blogs, and software vendor websites. However, one of the key features of lateral movement that makes it so difficult to detect is that for every individual malicious authentication that indicates malicious lateral movement, there are likely thousands of non-malicious authentications that indicate normal network traffic. Therefore, the process of log management and filtering out millions of unimportant events is as important a feature of lateral movement detection as is knowing the threat vectors available to attackers. Since this research was not conducted in an enterprise environment and simulating such an environment is costly, this element of the research was simulated to a small extent and would benefit from being run in an enterprise environment to prove its efficacy in production environments.

Definition of Terms

- Dynamic Link Library: A Dynamic-Link Library (DLL) is a library that contains code and data that can be used by multiple programs at the same time [8].
- Server Message Block: Server Message Block (SMB) is a network communication transfer protocol to provide shared access to files, printers, and ports between the networks [9].
- Remote Desktop Protocol: The Microsoft Remote Desktop Protocol (RDP) provides remote display and input capabilities over network connections for Windows-based applications running on a server. RDP is designed to support different types of network topologies and multiple LAN protocols [10].
- Windows Driver Development Kit: The Windows Driver Kit (WDK) provides a set of tools that you can use to develop, analyze, build, install, and test your driver [11].
- ShadowMove: ShadowMove is a modern stealthy lateral movement technique designed by students and faculty at the University of Illinois at Springfield, The University of North Carolina at Charlotte, and Louisiana State University. The code utilizes DLLs to execute lateral movement without being detected by traditional antivirus and without creating new authenticated sessions. The proof-of-concept code provided for this research was built as PoC.exe, and this executable name will be used synonymously with ShadowMove throughout the remainder of the research paper [12].

Lateral movement is the main problem that cannot be completely solved without vastly reducing business functionality. Methods used by attackers to move laterally are well documented, and prevention methods are readily available and accessible. However, lateral movement is also complicated to detect as it typically appears to legitimate user authentications. The objective of this study is to provide a literature review on research that outlines methods for differentiating between legitimate and malicious movement between computer systems, to highlight methods that have been proposed to prevent malicious lateral movement methodologies as well as the methodologies these prevention techniques aim to solve, and finally, to study a novel intrusion detection method and provide an original solution to detecting the method.

Background and Review of Literature

This literature review aims to provide the reader with a deeper understanding of the work conducted in lateral movement detection attempting to solve some of the major problems that exist in differentiating malicious traffic from regular traffic. This chapter will be divided into four sections; the first, another short outline of the nature of the problem; second, a review of literature related to the problem; third, a review of literature related to the methodology; and finally, a summary of the research done in preparation for the original study.

When approaching the problem of lateral movement, two primary questions are posited; how do we differentiate between malicious and benign traffic, and how do we stop malicious lateral movement? Differentiation and prevention of known techniques are the two areas in the background portion of this literature review. Since methods for lateral movement are well known and documented, this literature review begins with a review of the MITRE ATT&CK Framework. An overview of each method used for malicious lateral movement and proposed defenses are covered. A diagram outlining common tactics to prevent lateral movement is provided in Appendix A for engineers looking for easy solutions to cover most attack methods. A literature review proposing new detection types for more precise differentiation and remediation is provided following the MITRE review. MITRE outlines numerous attack methodologies for lateral movement in their ATT&CK framework. This section discusses all methods described and their recommended mitigation methods for fixing the vulnerabilities. The purpose of this section is to highlight common methodologies employed by attackers when moving through a Windows environment. The methods covered by MITRE are common exploits with well-known mitigations. In writing this section, this research aims to highlight that while extremely common in cyber-attacks, common mitigations using built-in protocols exist for every modern-day security practitioner.

Exploitation of Remote Services

Attackers exploit remote services to gain an initial foothold in the network but can also be used once inside the network to move between systems. A common example of remote service exploitation is outlined in CVE-2017-0143 or “Eternal Blue”. Eternal Blue is a vulnerability known by the NSA and released to the public once the NSA discovered that attackers were using the exploit maliciously in other environments around the world. Eternal blue uses a vulnerability inside Windows SMB and allows a remote attacker with no credentials to gain SYSTEM level privileges on the target machine.[13]. Common mitigation techniques outlined for this type of attack include sandboxing applications to discover vulnerabilities, uninstalling unneeded or unused services from all systems, installing exploit protection software that can stop an exploit when found, implementing a strong network segmentation policy, minimizing the permissions and access of all accounts through a privileged access management project, improving employee knowledge of threats and attacks through training, update all software to the latest and most secure versions, and frequently scanning the network for vulnerabilities with updated databases to ensure all vulnerabilities are patched as they become known.

Internal Spear Phishing

Internal spear phishing is when an attacker compromises an internal email address and uses it to gain the trust of other internal users to trick them into sharing passwords or other sensitive data. For example, attackers may create phishing campaigns with credential harvester pages or phish for information by emailing colleagues using a trusted address to gain information. Mitigating internal spear phishing attacks is extremely difficult as an initial breach has already occurred, and all attack traffic looks like standard email traffic. Employee awareness programs and training will help reveal internal spear phishing campaigns, but fully mitigating them is impossible without interrupting business systems.

Lateral Tool Transfer

Once inside a system, attackers will transfer tools from one system to another by exploiting administrative accounts, opening SMB file servers, network drives, or removable media. By transferring attack tools to other systems, attackers can connect to and create a backdoor on whatever system they place it on, giving them more profound persistence in the network. Common mitigations for this attack include filtering network traffic to ensure that only known devices and addresses communicate with secure channels like SMB or SSH. Another method for preventing lateral tool transfer is implementing a network intrusion prevention system. By implementing a signature-based or anomaly-based intrusion-prevention system, irregular traffic or file transfers may be detected and prevented.

Remote Service Hijacking

Attackers sometimes can hijack pre-existing network connections using services like SSH and RDP. Attackers may commandeer these sessions to act against remote systems like transferring files or executing commands. Detecting service hijacking is difficult since the authorized user creates the initial session, and the malicious actor does not create a new session. Likewise, mitigation is difficult as it relies on disabling features and services when unneeded, implementing a strongly segmented network, managing privileged accounts, and managing user accounts. Ensuring only accounts with the need to access the service can access the service will reduce the remote connection footprint and make it more difficult for the attacker to hijack a connection. ShadowMove uses a novel method for hijacking unencrypted sessions between computers on any port. This will be covered more extensively in section five.

Remote Services

Attackers will use compromised accounts to use services like RDP, SMB, SSH, and VNC to connect to remote computers. There are numerous ways for attackers to gain valid credentials on remote connection applications, including hash dumps, passwords left on files, brute force guessing, and many others. Mitigation of this threat vector includes implementing multi-factor authentication where possible and managing user accounts to ensure only the users that need access to the remote services are allowed to access the remote services.

Replication through Remote Services

To bypass air gaps or to increase the likelihood of reaching difficult-to-reach machines, attackers may copy malware to removable media in the hopes that it is inserted into another machine where they will have access to more sensitive data. Mitigations include disabling autorun as attackers have used the autorun feature to execute malware automatically when a user inserts the removable media device into a new computer. Likewise, limiting USB storage devices on networked computers will make it nearly impossible for removable media to be used as an attack vector.

Software Deployment Tools

Attackers may gain persistence on any number of machines by accessing applications that deploy software across a network. By compromising an account on Microsoft's System Center Configuration Manager or McAfee E-Policy Orchestrator, attackers can gain the ability to deploy any software to any system within the network. Depending on how the software deployment tool is configured, it may be possible for standard network accounts to have sufficient permissions to deploy applications anywhere in the network. Mitigating this attack vector is accomplished by ensuring systems are isolated correctly in the active directory, ensuring multi-factor authentication is in place for critical systems, segmenting the network to keep critical systems isolated from less secure systems, enforcing a strong password policy, managing privileged accounts with a Privileged Account Management procedure or tool, ensure that tools with the ability to deploy software are configured so that only signed binaries or specific binaries can be deployed, update systems to ensure patches are installed when they are needed, manage user accounts to ensure over permissioned accounts are not present in the environment, and ensure that users are trained in the policy and procedures for deploying software to remote systems. Each company and environment will have a different level of access needed to install applications on systems remotely, so mitigating a threat like this can be difficult. Some companies will also have custom software that they may want to push, which may be unsigned. Companies should ensure that if they are going to use a remote deployment tool, which tool fits all the needs for the types of software they will be distributing.

Taint Shared Content

Attackers may be able to move laterally by adding malicious files to shared locations on the network. These tainted items will typically contain instructions that allow the attacker to move laterally once an unknowing user executes them. Attackers often design these files so that the user's intended action is still executed so as not to raise suspicion. However, the malicious script will run and allow deeper network access. Mitigating shared content includes using an exploit prevention system, file and directory permissions for users that have access, and to identify potentially malicious software with detection systems and auditing/blocking the execution of such files with tools like Microsoft AppLocker or Software Restriction Policies.

Use Alternate Authentication Material

Attackers also attempt to access alternative authentication materials like Kerberos Tickets, Application Access Tokens, Authentication Tickets, or Web Session Cookies to bypass the password required to access the service. For example, using meterpreter shells or programs like Mimi Katz to dump credentials or active tickets and sessions, attackers can gain the ability to craft a token or ticket that the system will take in lieu of a password. Mitigating these types of attacks includes privileged access management to reduce the likelihood of lateral movement between systems and implementing a principle of least privilege within the network to mitigate the number of administrative accounts on the network.

Liu et al. discussed the problem inherent in Lateral Movement detection by outlining two key issues when differentiating lateral movement from normal use behaviors: detecting the path after discovering an infected computer. Latte analyzed large-scale event logs collected from operational networks [14]. Their system analyzed Kerberos service ticket requests to construct a graph outlining a general connection structure between networked machines. For general detection purposes, Latte uses this connection graph and data from Windows Event Logs to correlate rare connections in conjunction with Remote File Execution to detect possible lateral movement within an environment. To prevent log tampering, Windows system file logs are sent to the Windows Event Forwarding server and fed to MapReduce to create a complete historical map of remote file executions and Kerberos service ticket requests. The work done by Liu et al. stands out from other graph-based models in that it can be deployed to stock Windows installations as it only utilizes logs gathered from standard installations of Windows and requires no kernel-level privileges to operate as intended. Latte truly shines when trying to forensically analyze an attacker's path to and from an infected node and highlights useful information for future analysts investigating potential lateral movement attacks. By analyzing the known compromised node and filtering out the rarest results, analysts are only required to make a limited number of manual investigations to find paths taken by the attacker. In their experimentation, the forensic analysis module was able to successfully float the malicious paths taken by the attacker to the top eleven results out of a possible 447,828 paths [14]. Given their method, analysts need to manually analyze the eleven paths discovered by the forensic analysis module: a far more manageable task than the 447,828 paths in the first dataset. Since analysts in the eleven top results discovered the malicious paths taken between workstations, the researchers determined their forensic analysis module to be successful. The authors admitted that for general detection, relying solely on the rare node connections generates far too many false positives to be considered a practical source for actionable insight in an environment. In each ninety days, over forty-four million connections were tied as the most suspicious to generate fewer false positives, and the authors recommended first determining where remote file execution occurs within a network and then correlating the rare connection paths inbound and outbound from the system wherein remote file execution took place. This research, however, did not propose a method for how analysts can differentiate between malicious and non-malicious traffic. By correlating Kerberos Service Tickets with remote execution and analyzing rare paths using a network map, it is possible to narrow down the possible malicious lateral movement events to a level where an analyst can manually analyze each in a given workday.

Holt et al. researched the use of Deep Autoencoder Neural Networks in detecting lateral movement in networked computers. They begin by outlining that many other researchers have studied using neural networks to aid in detecting intrusions in computer networks. However, Holt et al. differentiate their research from past research endeavors by setting out to solve the problem of lateral movement rather than general intrusion detection. Holt et al. used the Los Alamos National Laboratory dataset to train and test their neural network. The Los Alamos National Laboratory dataset covers 58 days and is over seventy-three gigabytes in size. Therefore, Holt et al. used two subsets of data from the Los Alamos National Laboratory dataset: a developmental dataset for use in training and proof-of-concept work and a test dataset to evaluate the accuracy of their created models [15]. The developmental dataset included all the red team data from the Los Alamos Dataset and all normal traffic from the computers compromised by the red team.

Researchers added a random data sampling to make the developmental set more varied and avoid overfitting the data. Researchers created the test dataset similarly by adding all users from all compromised computers to add more variance to the dataset. After describing how unsupervised autoencoders learn, the authors describe four models they designed for testing. The first was a shallow model designed 6-2-6, the second was a deep model designed 6-3-2-3-6, the third was a deep model designed 6-3-2-3-4-5-6, and the fourth was designed 6-5-4-3-2-3-6 [15]. After feeding data to the neural network for testing, the results were mixed. The first three models performed well with low false positive rates--.55%, .85%, and .95%--with good recall, however, performed inaccurately in the precision metric. The fourth model had a false-positive rate of over 20% and no measurable precision nor recall. The three models proposed by Holt et al. performed worse than the semi-supervised model they reference in their related works section. However, the semi-supervised model proposed by Siadati et al. [16] requires a human analyst to aid in the detection of anomalies and is not fully automatic like the model proposed by Holt et al. Furthermore, the model proposed by Holt et al. was more accurate than the model proposed by Bohara et al. [17]. While the results show positive progress towards automating intrusion detection and lateral movement detection using autoencoders, further research must be conducted to improve the detection rates and reduce the volume of data necessary to train an autoencoder to perform intrusion detection. Intrusion detection using machine learning is a critical area of research, and numerous researchers investigated the use of unsupervised and semi-supervised machine learning approaches to aid in the filtering of data to a manageable level or to work as IDS/IPS in the network [14][18][19][20][21]. Many semi-supervised models perform extremely well when pairing the judgment of a human with the pattern recognition of a computer [22]. Methods researched by teams like Holt et al. showed promise in automating tasks and reducing the amount of noise while more accurately predicting abnormal user behavior as is presented during a malicious lateral movement event.

Abdurrahman Pektas and Ertugrul Basaranoglu introduced a new method for conducting penetration tests within a Windows Environment. They claimed that there has not been a structured attack method for Windows penetration tests and set out to construct a new method that focuses specifically on attacking Windows environments [23]. The authors began their article by outlining the basics behind other penetration testing methodologies introduced by companies like OWASP and the CE-Council but quickly started working on demonstrating why their Microsoft Domain Environment Penetration Test Methodology is superior for testing Windows environments. The authors introduced a ten-step systematic process for attacking Windows environments. They explained methodologies used throughout the penetration test within each step to gain access, attain persistence, and compromise more systems. Section three of the paper introduces numerous methods for attacking Windows environments and explains methodologies that attackers use to breach a Windows environment successfully. The authors break down their methodologies in the ten-step penetration test method. Section four covers mitigation techniques for preventing unauthorized access to systems as laid out within section three. While comprehensive in scope, the amount of detail in preventing specific methodologies is lacking. While this paper introduces a new structure for attacking Windows environments and the mitigation is a minor portion of this attack framework, a more comprehensive list of mitigation techniques for the numerous specific attack techniques would have been helpful. The authors' concluding section outlined that since they provide more steps, specific tools for attack and mitigation, and different techniques, their method competes with other attack methodologies for conducting penetration tests. It is true that system administrators and security professionals could use this framework to aid in penetration tests and securing their Windows environments. However, for this starred paper, this resource helps outline novel methods for exploiting Windows environments for lateral movement as well as potential measures to prevent lateral movement. Many of these activities are also outlined in the MITRE ATT&CK Framework and will be covered in future sections. This research helps develop a broader understanding of tools and techniques available to network defenders and how malicious lateral movement may be defended against.

Bai et al. proposed a new method for classifying RDP sessions in Windows environments. Using datasets from the Los Alamos National Laboratory and supervised machine learning algorithms, the authors proposed a new method for detecting and sorting through RDP sessions to better classify malicious lateral movement within a Windows environment. Bai et al. concluded their research by comparing their developed method to state-of-the-art methods and gauging their effectiveness based on another model's performance [24]. The authors began their research with a literature review of other authors that have tried to classify malicious RDP sessions using the Los Alamos National Laboratory Dataset (LANL) machine learning algorithms. Bai et al. critique Ussath et al.'s method [25] for being unwieldy in production environments, although the learning algorithm was efficient at detecting malicious RDP sessions.

Furthermore, the authors critique Kaiafas et al. [26] for their proposed use of the LANL dataset and posture that the LANL dataset is only useful for machine learning training when combining the two available datasets rather than solely utilizing the comprehensive events dataset. Bai et al. level criticism at the LANL dataset for its fractured nature. The comprehensive events dataset holds diverse red-team activities. However, the ratio of red team activities to normal activities is extremely small. Furthermore, the red team activities are launched from four machines and occur during specific timeframes.

For this reason, Bai et al. concluded that using the comprehensive dataset alone for training machine learning algorithms will lead to overfitting or training the machine learning algorithm to detect specific timeframes and machine ID rather than generalized patterns in the malicious RDP activities [24]. To solve this problem of overfitting the training data to specific activities generated by specific machines at specific time intervals, Bai et al. proposed combining two datasets from LANL to create a comprehensive dataset that combines more user events from the Windows event log with the malicious red team data from the comprehensive dataset to make a more generalized dataset to train machine learning algorithms and bypass the issue of overfitting by using only one data source [24]. Using their new combined dataset, Bai et al. test their training data on five different machine learning algorithm classifiers and determine their performance by measuring their accuracy, precision, recall and F-Score: the "harmonic mean of precision and recall" [24]. The authors then compared their model to another top-performing model proposed by Kaiafas et al. [26]. Using their dataset, Bai et al. were able to reduce the number of inputs, abstract the data more completely than Kaiafas et al., and return higher detection rates. In doing so, Bai et al. reported that their model is more useful in a production environment as it requires fewer data to run and is as effective as the more complex model [24]. This model helps highlight what Windows Event Log events can be used in automated systems to detect malicious lateral movement in an environment and highlight the fact that this task can be automated with sufficient training data. Understanding that Windows Event Logs can be used in automated systems to assist with detecting malicious lateral movement is a critical point of this research. Windows Event Logs are often overlooked as being clunky or not verbose enough. This research proves that Windows Event Logs can be utilized effectively for intrusion detection when the correct filters are applied, and careful logic is used. The machine learning algorithm proposed by Bai et al. demonstrated novel methods for detecting and preventing lateral movement using common tools accessible to most security analysts and engineers.

Hossein Siadati and Nasir Memon introduced a method for detecting anomalous logins and lateral movement within an enterprise network by creating a "network login structure" that outlines typical sign-ins for users and then employed an anomaly detection system to detect out-of-character logins for users within the network [27]. Siadati and Memon focused on credential-based lateral movement during which the attackers steal valid user credentials through tactics like pass the hash and authenticating as valid users. These attacks are some of the most difficult to detect because they closely resemble normal account authentications during an average workday. Siadati and Memon created a system that simply looks for odd login behavior from users rather than specific attack methodologies. By focusing on a broader scope, their method should be able to watch for a wider range of attack vectors. Siadati and Memon employed a pattern miner and login classifier to collect as much data as possible about typical user behavior in the network and classify whether the logins are considered benign or malicious, given the data mined by the pattern miner. Siadati and Memon created an algorithm to classify typical user behavior based on the login pattern, occurrence, orientation, patterns, and scores generated by all previously stated inputs [27]. Once researchers completed their system, they evaluated their detection system against a dataset holding five months of data from a global financial company. Once the test was run against the system, the data was handed to a group of analysts from the company, and each flagged instance was investigated to determine whether it was a true positive or not. The analysts, after analyzing the flagged sign-ins, discovered that the system only had an 11% accuracy rating. This was because administrative logins tend to look abnormal in many cases as administrators constantly access new machines for the first time, causing the pattern miner to flag them as malicious given their infrequency. While monitoring standard behavior for users and flagging anomalous logins is a good theory, in practice, more information must be considered before flagging anomalous logins as malicious. For instance, taking process history from the user before the connection was made or observing spawned processes after the connection was completed could help narrow the scope and improve the system's overall accuracy. While some sign-in-based anomaly detection system could help detect novel lateral movement techniques, further studies into this subject need to be done before this type of detection can rely solely on malicious lateral movement detection. While not the most effective solution, detecting lateral movement by tracing anomalous logins is a worthwhile endeavor in a defense-in-depth structure. It is another method by which analysts and engineers may detect lateral movement within the infrastructure.

Kaiafas et al. aimed to solve the problem with anomaly detection outlined in the paper by Siadati and Memon: false-positive detections. Kaiafas et al. tried to solve this issue by providing more contextual data surrounding the authentication of the classifiers [26]. By including more contextual data, they aim to reduce the number of false positives by classifying what normal behavior looks like more accurately. Kaiafas et al. used four different supervised anomaly detection systems in their research and tested their accuracy using the Los Alamos National Laboratory Dataset. Since the Los Alamos National Laboratory Dataset has so few malicious activities—less than .00033% of total authentication logs [26]—filtering the anomalous/malicious traffic from standard user traffic is extremely difficult. To assist their supervised learning algorithms with sorting malicious events from non-malicious events, the authors identified several features and included tangible pieces of data to improve malicious anomaly detection. The first feature is the "distribution of time difference of events between systems and from user to system" [26]. This feature captures the spread of user activity over time, allowing the detection engine to estimate a relative pattern to user activity. The second feature is "user activity and connection frequency" [26]. The authors used this to estimate a general pattern of typical user behavior on a given day. By observing the frequency of network activities, the pattern recognition system can better find whether actions taken by a specific user account are outside the normal range. The third feature is the "distribution of malicious events if we see every event as a trial" [26]. In their experimentation, Kaiafas et al. supplied a probability to the anomaly detection engine, which outlines how likely a malicious event is. While this is helpful in an experimental system, when moving to an enterprise network, this number will not always be known. The fourth feature is "user variance" [26]. This feature outlines the significance of a user during a given period and is designed to tell the system how often a specific user should be expected to authenticate. It creates a distribution of both the number of users authenticating during a period and the expected spread of user activity, meaning the more popular users should be expected to authenticate more frequently during a given period.

After running the dataset through these classifiers, the authors fed the data to four different "ensemble learning techniques" [26] for final classification. These ensemble learning techniques use multiple machine learning algorithms to classify and sort data. They used LogitBoost, Random Forest, Logistic Regression, and Majority Voting. After training their systems with a subset of data from the Los Alamos National Laboratory Dataset, Kaiafas et al. measured the success of their systems by computing the false positive rate, false negative rate, and balanced accuracy, which is "the arithmetic mean of True Positive Rate and True Negative Rate" [26], Positive Predictive Value: a ratio of known malicious activities vs predicted malicious activities, the F1-measure, and the Prevalence: or the ratio of True Positive and False Negative over the sample size. After conducting their tests, most models performed well with low false positive rates, with the Majority Voting system outperforming the others by a small margin. The systems achieved a 0% false positive rate for 68% of the data and a .0019% false positive rate for the remaining 32%. The authors conclude that completely avoiding false positives is a fool's errand, however, minimizing the number of investigations made by human analysts is the goal of most semi-automated systems. Finally, Kaiafas et al. proved that their sorting methods effectively reduce noise generated by the administrator. This research is fundamental in feature choice for reducing the noise generated by network logs. Kaiafas et al. supplied many features that reduce the false positive rate generated by network logs. The downside to this method is that the ratio of benign authentications to malicious authentications is known. It would be interesting to see how a system such as this would perform in a black-box environment.

Ussath et al. investigated twenty-two different APT attacks to gather the best practices used by many APT's to attack networks. In doing so, Ussath et al. proposed highlighting better detection methods for commonly used attack structures [25]. To simplify the complexity of APT attacks, Ussath et al. viewed three main categorizations of activities taken during an APT campaign: initial compromise, lateral movement, and command and control [25]. After the first compromise, the authors explain the importance of lateral movement in computer systems for all the APT groups. The most common method for moving laterally through systems found by the authors is to use pre-installed Windows tools like remote desktop protocol, windows management instrumentation, PowerShell, and PS Exec [25]. In addition, attackers often collected passwords from memory using tools like Mimi Katz or Windows Credential Editor. Attackers rarely brute-force passwords as brute force attacks are noisy and are typically prevented by administrators. The final method outlined for lateral movement by Ussath et al. was to exploit known vulnerabilities to elevate privileges. The authors proposed that attackers exploited vulnerabilities because access to passwords and password hashes required administrative credentials [25]. To detect malicious lateral movement, Ussath et al. proposed detecting known malicious processes like Mimi Katz for password and hash dumping activities as well as monitoring the Local Security Authority Subsystem Service process, which has direct access to the memory of other processes and is a vector of attack for dumping credentials [25]. Viewing the chart of twenty-two different APT groups created by Ussath et al. provided a good snapshot into the processes and attack methodologies used by APT groups. Understanding the methods used by APT groups and common defenses against them helps with understanding how to detect and prevent attacks. The table created by Ussath et al. is provided in Appendix A and outlines the most common methods used by APT groups and provides a good overview of attacks to focus on defending against.

Literature Related to the Methodology

ShadowMove: A Stealthy Lateral Movement Strategy

Niakanlahiji et al. proposed a novel lateral movement strategy that uses built-in Windows features to jump between systems using existing connections while bypassing all modern AV detection [12]. The system works by duplicating socket connections and hijacking established FTP, TDS, and WinRM connections. The system proposed by Niakanlahiji et al. used three main steps: Duplicate a socket used by a legitimate client, inject packets into the TCP stream using the duplicated socket, and spawn a new session of ShadowMove on the server handling the packets by tricking the server into executing the injected packets. This novel method for lateral movement can avoid detection because it only reuses pre-established connections and never spawns a new connection with the server, thereby not generating a new TGT or TGS request as is typical in standard lateral movement attacks. The initial breach requires that a piece of malware be installed on the initially infected vector. However, given the stage at which lateral movement takes place during a cyber-attack, it is believable to assume that the attackers would have created a layer of persistence on the systems and had a way to deliver a malicious payload to the client. The ShadowMove software has six modules: Connection Detector, Socket Duplicator, Peer Handler, Network View Manager, Lateral Movement Planner, and Plan Actuator. Each module has a specific purpose during the lateral movement phase of a cyber-attack, and each serves a unique purpose in helping ShadowMove function as intended. The Connection Detector is a listener that waits for a change in status from non-established to established and records when a certain TCP port is used. This system constantly queries the TCP table on the Windows machine to find when a vulnerable port has an established connection. The peer handler is used to share data between instances of ShadowMove within a network. Using duplicated sockets, process suspension, and previously compromised sockets, the peer handler can communicate with other ShadowMove instances to share knowledge about the architecture of the network. The network view manager is a dashboard from which the attacker can view the status of the network that has been compromised thus far. The attacker can view hosts, sockets that have been duplicated, IP addresses, ports, service types, and other valuable information the attacker may want to know when engaging in lateral movement as part of a cyber-attack. The Socket duplicator duplicates sockets. This is done on a Windows system by using an open process to enumerate all open handles. Then using "GetPeerName" it enumerates the socket from the AFD handle. Finally, it uses "WSADuplicateSocket" to duplicate the socket, giving the attacker a tunnel from which to inject packets into the data stream. Since these packets are injected into a data stream where the benign application is running and transferring data, ShadowMove uses "SuspendThread" to pause the execution of the benign service in order to ensure its own code is injected and executed. The lateral movement planner gives the attacker the capability to view an exploit map and plan for the most efficient lateral movement attack. Since permissions between systems vary in a Windows environment, not every connection will have permissions to read, write, and execute on other systems. The lateral movement planner shows the attacker the best route possible to a given target and can plan the most efficient route to reach the desired system. Finally, the lateral movement actuator contains modules responsible for crafting and reading from packets midstream and for crafting packets that can hijack FTP, MS SQL, and WinRM connection streams. Niakanlahiji et al. created a stealthy lateral movement technique that bypassed all traditional antivirus, endpoint detection and response tools and IDS/IPS tools leveraged against the ShadowMove software. However, the authors outline a few key issues with their design. First, enabling protected processes would stop ShadowMove from duplicating the process handle. Second, the ShadowMove architecture only works on unencrypted channels: thereby limiting attack vectors to specific protocols in a network. However, the novel method by which ShadowMove jumps from system to system effectively bypasses antivirus and endpoint detection and response systems. This makes it a prime candidate for attackers to improve on and make lateral movement attacks in less distinguishable ways. This method will be expanded upon in sections III, IV, and V, as this research aims to invent a novel method for detecting this ShadowMove attack.

Detecting Adversary using Windows Digital Artifacts

Seng Pei Liew and Satoshi Ikeda proposed a method for detecting advanced persistent threat adversaries in a Windows environment using nothing but native Windows artifacts [28]. The authors started by outlining two key issues with detecting persistent adversaries in a Windows environment. The first issue was that attackers use benign file names or files to conduct attacks to prevent signature detection. The second was that disparate configurations within Windows environments and the lack of conformity to a standard make tracing paths difficult. To overcome these issues, the authors proposed a machine learning-based approach that observes digital artifacts left in all Windows systems. To do this, the authors also proposed a new algorithm to learn the execution time of a process from the shipmate [28]. Using the data gathered from the Shimcache and the output of the machine learning algorithm, the authors proposed an adversary detection system that, given a period, would return a score representative of how malicious the behavior taken during the given time period was. The authors outline their approach to detecting APT's within an environment. By breaking down the attack pattern of APT's to component parts, the authors outline the Windows commands that can be run during an attack.

Assuming a breach has occurred, the authors' outline commands typically run during the persistence, discovery, privilege escalation, lateral movement, defense evasion, and exfiltration phases of an attack. Given some of the most common commands used during an attack, the authors explain the digital artifacts created by running the tools in the Master File Table, Shimcache, Prefetch, and Windows Event Log during execution. The authors explained their methodology for tracing an attack using these event artifacts. They outlined their algorithm for determining the execution duration of a file using artifacts found in the Shimcache: a proxy between Windows versions that ensures backwards compatibility of executables [28]. After explaining the details of the timing algorithm, the authors explained how their machine learning-based scoring algorithm could aid in detecting malicious behavior in Windows environments. Using inputs from the Shimcache, Prefetch and Windows Event Logs, the machine learning algorithm computes the data and scores the time frame accordingly. As outlined above, the scoring module takes a list of commands commonly used by attackers to execute distinct phases of an advanced persistent threat attack [28].

The algorithm used for training is a Random Forest algorithm, a black-box training method. This means that the researchers know the data they put in, but the computations on the data inside the algorithm are unknown to researchers. They found that implementing the model in this manner gave them a precision of 86.7% and a recall score of 75.6% [28]. The results are not fantastic, and researchers were upset that specific applications like PowerShell were flagged as malicious even when other processes were not spawned from the parent process. Part of the issue with the method is that the researchers only focus on a small slice of application execution, by only focusing on a small number of applications, processes, and indicators of compromise.

Furthermore, researchers only supply the machine learning algorithm with a narrow slice of time and decide on malicious behaviors that took place over a long period. As a research piece, it is interesting to note how a machine learning-based model with basic Windows events can have some success at detecting malicious behavior in a Windows environment. However, universally applying these rules to a networked environment would not give sufficient data to analysts looking to protect a production network. The most helpful research conducted in this study is the use of default artifacts inherent in all Windows systems to assist in the detection of malicious behavior in an environment and could be used in numerous other approaches to reduce the need for specialized endpoint monitoring systems to be installed on user workstations. What is important to note, however, about this research is that the Windows operating system creates enough logs and artifacts to successfully identify malicious behavior without the use of third-party applications. A similar methodology will be employed in sections III, IV, and V as this research attempts to detect ShadowMove.

Detecting Abuse of Domain Administrator privilege using Windows Event Log

Fujimoto et al. compared methods for detecting the abuse of domain administrator credentials proposed by other researchers. Since many detection methods are interested in detecting specific CVE's and attack methodologies like "Mimi Katz" or "Kerberoasting," the researchers are interested in combining the eclectic methodologies into a central repository of detection methods that can be used to detect abuse of domain administrator credentials into a single tool [29]. The researchers outline useful methods proposed by other researchers to detect abuse of domain administrator credentials. A detection method proposed by Shingo Abe outlines using Windows Event Logs to detect abnormal administrative access to resources by correlating historical data with daily use of administrative credentials [30]. Fujimoto et al. include research done by Shusei Tomonaga at JPCERT/CC into common commands executed by attackers during an APT campaign [31]. Fujimoto et al. focused solely on correlating Windows Event Log 4688−A New System Process Has Been Created−to detect abuse of domain administrator privileges. Fujimoto et al. use research by Junghoon Oh at AhnLab to detect APT lateral movement using administrative shares to spread access

[32]. Fujimoto et al. use the event log 5140−A network share object was accessed−to determine if an administrative account has wrongfully accessed a network share: a common tactic used by attackers to spread malware across the domain. Finally, Fujimoto et al. include research done by Idan Plotnik et al. and Andrey Dulkin et al. to detect golden ticket creation by logging Kerberos Service Ticket requests that have no prior Ticket-Granting-Ticket (TGT) associated with them [33][34]. Fujimoto et al. considered all these known methodologies for detecting abuse of domain administrator accounts and developed their method with a high detection rate. Their method focused on watching the domain controller for the creation of golden tickets or credential theft and did not detect abuse of all machines in the domain. This method, therefore, is not used to detect lateral movement wherein the attacker does not contact the domain controller for escalated privileges: e.g., in the case of spear-phishing an escalated account. Fujimoto et al. propose a sophisticated signature detection system that utilizes built-in Windows Command-Line-Interface (CLI) tools and known privilege escalation methodologies to detect APT privilege escalation and Domain Administrator account abuse in a Windows Active Directory environment. Their results are subpar as they have a high rate of false negatives across all detection categories.

Furthermore, not detecting abuse of a domain administrator account typically means that the attackers have compromised the entire domain, and quick and exact remediation must occur at once. While novel in its scope and thorough in its investigation of methodologies used to compromise domain administrator accounts, different methods must be used to detect and prevent privilege escalation attacks more precisely. The important artifact from this research is that only Windows event logs were used to detect malicious movement in a system. Nevertheless, it is possible to garner valuable information from intrinsic logging sources.

Overall, whether proposing new methods for detection like Holt et al., Lie et al., or Bai et al., or presenting methods for preventing known attacks like the MITRE Group or Pektas et al., this literature review was designed to give the reader a flavor of a wide swath of research that is ongoing in the field of lateral movement detection. It covers new methodologies for improving detection rate, reducing false positives, and increasing lateral movement defenses by using graphs, machine learning as well as new frameworks of thought. Finally, by outlining ShadowMove and using Windows event logs and digital artifacts to detect lateral movement, the final section of the literature review was designed to give the reader a strong base of knowledge from which to draw when reading about the original methodology and tactics utilized in this research to detect ShadowMove. This feat has not yet been completed.

To bypass air gaps or to increase the likelihood of reaching difficult-to-reach machines, attackers may copy malware to removable media in the hopes that it is inserted into another machine where they will have access to more sensitive data. Mitigations include disabling autorun as attackers have used the autorun feature to execute malware automatically when a user inserts the removable media device into a new computer. Likewise, limiting USB storage devices on networked computers will make it nearly impossible for removable media to be used as an attack vector.

Methodology

Lateral movement is a well-documented strategy employed by attackers going after enterprise systems. In recent years, defense and detection methods implemented by defenders have gotten more sophisticated. Enterprise tool sets from companies like Stealthbits, Arctic Wolf, Rapid 7, Palo Alto, and others have the capability and built-in parameters to detect traditional lateral movement techniques like Kerberoasting, Pass the Hash, and Pass the Ticket. Likewise, many of these toolsets also include anomaly detection capabilities that monitor user activity, create a baseline for typical use, and alert when the user strays outside the normal boundaries of daily activity. For traditional lateral movement techniques, these detection methods are more than enough to determine whether a compromised user or an attacker-created user is bypassing security protocol and moving abnormally through enterprise systems. Once attackers gain a foothold during an attack, the traditional methods for expanding influence within the network include remote service exploitation, tool transfer, session hijacking exploiting remote services like SMB or RDP, replication through removable media, software deployment tools, shared-content poisoning, or alternative authentication material usage like pass-the-hash or pass-the-ticket. Many of the toolkits and methods for dumping credentials or copying hashes like Mimi Katz or LSASS dumps are detected by traditional antimalware companies. In many cases, an inexperienced attacker using standard toolsets like Mimi Katz will be caught by traditional endpoint detection since many of these programs are picked up and deleted based on signature or behavior. In this section, we study a proof-of-concept code that utilizes packaged DLLs, hijacks ongoing TCP connections, and enables the attacker to move laterally through Windows systems undetected by endpoint protection software. In studying the code, we determine a framework to audit, detect, and alert on possible execution of lateral movement using the Shadow Move lateral movement variant. While this case study focuses on a single malware variant that hijacks specific DLLs, the process for enabling more advanced Windows auditing, parsing high volumes of data for relevant audit logs, and narrowing the scope of the search for specific DLL function calls is highly relevant to any malware strain that may use similar methods to the Shadow Move attack variant.

VM Configuration

The test environment was created in VMWare Workstation Pro on a host machine running Windows 10 Pro. The virtual machine was given 8 Gigabytes of RAM, two processor cores with two threads each giving it four logical processors, a 120 Gigabyte hard disk, a network card using NAT, and two monitors using 3D acceleration. There was also a folder shared between the host and the guest operating system to move files between systems for an easier research experience.

Software Used to Build the Binaries.

Windows 10 was installed on the virtual machine and updated to version 10.0.18363 with all required and recommended patches. To build the ShadowMove code for testing, Microsoft Visual Studio 2019 Community Edition—version 16.9.2—was installed along with all the C, C#, and C++ packages for Windows development. A screenshot of the installation options used can be seen in Figure 1.

Finally, the WDK is used to access the ntdll.lib and Ws32_32.Lib files for the ShadowMove build [35].

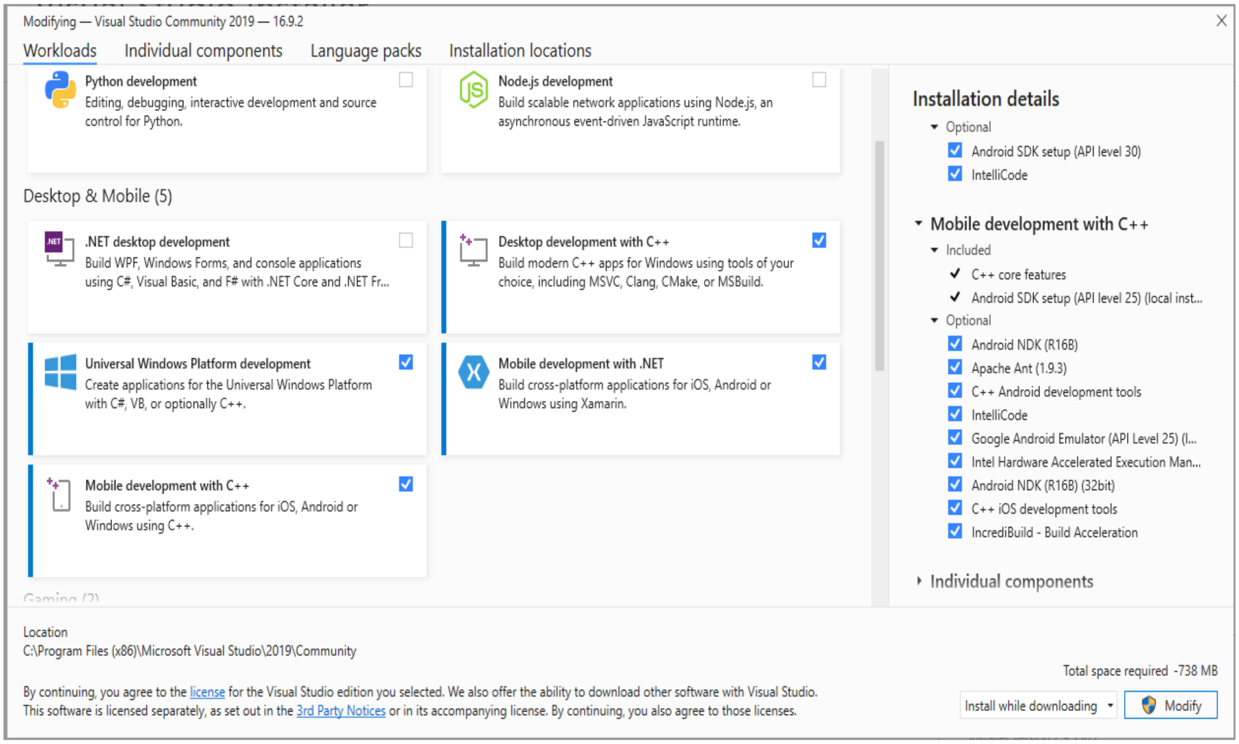

Figure 1. Packages installed to build C, C#, and C++ code in Windows 10.

Software used for monitoring, diagnosis, and alerting.

To monitor the software and DLL function calls at runtime the software API Monitor v2 was utilized [36]. This program monitors the application stack and detects which functions are called from a specific DLL during a software's runtime. To monitor Windows Event Logs generated at runtime, a free trial of DataDog—a cloud-based SIEM tool that can handle a wide variety of logs—was used to monitor, filter, and search the logs generated on the virtual machine [37].

ShadowMove Essentials and Socket Duplication

A group of faculty and students from the University of Illinois Springfield, the University of North Carolina at Charlotte, and Louisiana State University developed an attack methodology that utilizes Windows DLL and API functions to move laterally between Windows systems without the need to steal credentials or generate new authenticated sessions. The team named this stealthy movement technique ShadowMove. The code functions by exploiting normally trusted connections between any two networked Windows devices running unencrypted connections like FTP, WinRM, and MS SQL. The team released their research in August 2020. Typical signature and anomaly-based malware detection software look for custom executable code that either has patterns that are known to be malicious or have a signature of a known malicious file. By using code that calls internal Windows functions, the team was able to bypass standard antivirus programs because the operations done by the executable are standard functions that the Windows operating system depends on for normal use. Likewise, using trusted Windows DLL ensures that the attacker can run the malware on any modern Windows system so long as the libraries are updated sufficiently. Finally, when security analysts search for lateral movement, one of the key giveaways in traditional methods is the creation of new authenticated sessions using legitimate credentials or the dumping/stealing password hashes from Kerberos or LSASS. Once the attacker gains an initial foothold on the system, no credentials need to be stolen from any location, nor do any new authenticated sessions need to be created between systems. Since the attack takes place on authentically generated sessions by the authorized user and the ShadowMove code only latches onto established sessions, it is extremely difficult for an analyst to discover the lateral movement when following traditional attack patterns. To understand how ShadowMove hijacks a specific socket duplication event in the Windows operating system, we must first understand how the system establishes a typical socket duplication event to share a socket with a remote program. First, a local or host process will call the WSASocket API from the WS2_32 DLL. This, in turn, calls NTCreateFile, creates a new SOCKET_INFORMATION object, and calls NtDeviceIoControlFile, which creates kernel-level information about the handle. Second, the host process will call WSADuplicateSocket from the WS2_32 DLL to duplicate the socket and share it with the guest process. WSADuplicateSocket will copy the data stored in handle 1 and create a copy called handle 2. Finally, the guest process will call WSASocket to extract handle 2 and use the information contained therein to call NtDeviceIoControlFile to retrieve the same kernel-level information placed by the host program in step 1. Once this is complete, both handles share a duplicated socket, and the host and guest process can communicate using the same socket. ShadowMove interrupts this process by injecting itself into this workflow by copying handle 1 from the host process into a new handle 2, and using the copied object to connect to the socket in a similar fashion as a normal guest process would. However, ShadowMove differentiates itself from the standard guest process as it uses the ntdll DLL for querying system information and duplicating the system object. ShadowMove, in its most basic form, takes place in five steps. First, it calls NtQuerySystemInformation from the ntdll DLL to query system information and find a handle that it can copy. When it finds a handle to copy, the program determines whether the object it is attempting to copy is an ancillary function driver (AFD). Second, if the object it finds is an AFD handle, ShadowMove calls NTDuplicateObject from the ntdll DLL and creates a copy of the original handle. Third, ShadowMove queries the peer name passing the handle as a parameter. The handle duplication is bypassed until a peer name is found matching the name in the handle. This is to ensure that the connection that is hijacked is the one between the two desired peers. Fourth, when the correct connection is discovered, ShadowMove calls WSADuplicateSocketW from the WS2_32 DLL, passing the copied handle as the parameter. This creates an expected protocol structure that the kernel system on the host machine will expect. Finally, ShadowMove calls the WSASocketW API from the WS2_32 DLL, passing the WSAProtocol created in the previous step as the parameter. This step opens a duplicated−or shared−socket with the host machine and creates an injectable tunnel wherein ShadowMove can inject any data between the host and the guest without generating a new authenticated session.

Design of the Study

The ShadowMove proof-of-concept code was provided to by Md Rabbi Alam and Dr. Jinpeng Wei at the University of North Carolina at Charlotte. The proof-of-concept code comes in three parts. The first is a TCP Echo Server application written in C#. It is a simple TCP Echo Server that receives a string of text from a TCP Echo Client and returns the same text to the Client as was received by the Server. The second portion is a TCP echo Client that sends a message to a TCP Echo Server and receives the TCP echo reply from the server. The final portion included with the package is the ShadowMove proof-of-concept code which is written in C++. This proof-of-concept code works as described above, with the main caveat being that the ntdll DLL and WS2_32 DLL are packaged into the C++ executable using the C++ linker functionality in Visual Studio 2019, the ntdll.lib and the WS2_32.lib files found in the Windows Driver Development Kit [35]. The fact that the DLL files are linked internally with the binary after the software is built makes auditing specific Windows files significantly more difficult. After some basic troubleshooting, the ShadowMove code was compiled on the research lab virtual machine and was able to successfully duplicate the handle during runtime of the TCPEchoClient and TCPEchoServer. To detect specific API calls from within an executable at runtime, a process called hooking is required. One of the most well-documented and trusted free API hooking software available is API Monitor. Using API Monitor We were able to monitor all API calls from the PoC.exe ShadowMove code and find all instances of ShadowMove functioning as intended and duplicating a process handle. This is a crucial step, because DLLs contain numerous functions and determining exactly which function is called from the library is essential in determining if ShadowMove took place or another benign process was accessing similar libraries. There are native options in the Windows operating system to monitor DLL files. The Windows Security Auditing suite in conjunction with the Windows Event Viewer can give a security analyst or systems administrator the ability to view access to specific DLLs like WS2_32 or ntdll. However, the DLL files include numerous functions that the Windows operating system needs to function [8]. Therefore, monitoring the DLL file that contains the functions used for ShadowMove and alerting when the DLL files are called in a specific order is a way to give an alert that ShadowMove occurred, however, there is the possibility that this will generate many false positives as the operating system uses these files for standard procedures. Therefore, one of the greatest difficulties in alerting on the possibility of a ShadowMove taking place inside the operating system is monitoring the DLLs for specific function calls. In my research, we did not find a Security Information and Event Management (SIEM) solution that had the ability to monitor specific API calls from DLL files. Furthermore, since the C++ code links the ntdll.lib and WS2_32.lib files with the executable, the DLL files used for ShadowMove are called directly from the executable. This makes detection of the ShadowMove even more difficult. However, there are Windows Security Events that are logged by the Operating System that take place when ShadowMove occurs. Likewise, some events around the execution of ShadowMove also generate Windows Security Events. Since monitoring the DLL files is not always possible as they are linked, monitoring the Windows Events is the first line of defense in detecting ShadowMove. The methodology of this study includes compiling a functioning version of ShadowMove, running the attack against the TCP Echo Server and TCP Echo Client running on the same machine, monitoring the API Monitor Software to determine whether a successful ShadowMove socket duplication occurred, customizing the Windows event logs so that pertinent data is sent to the DataDog Cloud SIEM, monitoring the logs and creating customized views in DataDog to remove unimportant log files, and exporting a comma separated values file so an analyst can manually determine whether a ShadowMove may have occurred.

Data Collection and Tools and Techniques

To collect pertinent data to detect and predict when ShadowMove may have occurred, Windows 10 settings were adjusted to increase visibility into operating system events, and additional software was installed to collect the Windows Event Logs and parse the data once it was collected. Because ShadowMove utilizes Windows DLLs as part of its core functionality, auditing and monitoring the DLLs used by the malicious code is vital in determining when a ShadowMove may have occurred. For this, the local security policy was adjusted in Windows 10 to log file access and process tracking events in the Windows Event Viewer. To activate the necessary local security policies, open the Local Security Policy application by searching " Local Security Policy " in the Windows 10 search box and open the program. In the navigation menu on the left panel, expand "Local Policies" and open the "Audit Policy" subfolder. Within the audit policy subfolder, there are two important auditing policies that must be activated. The first is "Audit Object Access", which creates an event when a user accesses an item like a file, folder registry key, printer, or other items [38]. This policy is important because it registers events related to the closing of object handles. The second policy that must be activated in the local security policy is "Audit Process Tracking". This security auditing policy detects when a handle to an object is duplicated and when processes are started or terminated [38][39]. ShadowMove utilizes ntdll.dll and WS2_32.dll to duplicate and inject into non-encrypted network transmissions. To detect software that is accessing these specific DLLs to alert on potentially malicious handle duplication, auditing the access to the files is a function built into the Windows operating system. To activate the auditing feature on these specific DLLs, navigate to the files in the C:\Windows\System32\ folder, right-click on the file to be monitored, click properties, click on the " Security" tab, click on the button labelled "Advanced", click on the "Auditing" tab, click "Continue " to provide administrative privileges, click "Add", click "Select a principal", type "Everyone" into the box, click "Check Spelling", click "ok", select the check box next to the "Full Control" label, click "ok", click "Apply", click "ok", and click "ok". Once this set of steps is completed, anytime the file is executed, read, written to, or changed, an audit log will be sent to the Windows Event Viewer. This process should be repeated on ntdll.dll, mswsock.dll, and WS2_32.dll. The logs will contain timestamps, the user that accessed the file, and the process that called the file. Suppose ShadowMove is utilizing the DLLs packaged with the Windows operating system. In that case, when it touches a file during execution, the access will be logged and searchable by the SIEM tool or in the Windows Event Viewer. There is one final DLL worth mentioning that should be monitored for access. Since the version of ShadowMove that was run linked the ntdll.lib file and the WS2_32.lib into the executable using a linker function in Microsoft Visual Studio. The ntdll.dll and WS2_32.dll on the operating system were not touched during the execution of ShadowMove because they had them packaged into the executable. There is a file in the C:\Windows\SysWOW64\ folder called wshqos.dll. This DLL is called whenever an executable looks to access a function from a linked library. Since the more advanced version of ShadowMove uses linked libraries to hide its execution and intentions better, monitoring the wshqos.dll for access will alert an administrator whenever a file using linked library files is executed. To monitor DLL access and function calls during runtime to understand exactly how it functions, a program called API Monitor was installed to hook the DLL calls and monitor which APIs were accessed during the application runtime. Screenshots of the API Monitor software detecting the four main stages of ShadowMove are described in section IV.

Finally, to aggregate logs and implement a better search function, DataDog Cloud SIEM was utilized to collect all Windows Security and Application Logs from the Windows endpoint using the DataDog agent. The agent installation is documented on the website but simply requires the executable to be run by an administrator, the API key provided for the specific DataDog instance is inserted during the installation, and the log handler is installed directly from the DataDog Client Management Console using a few clicks. [37][40].

As most of the application code required to run ShadowMove is built into the Windows operating system, all the software needed to detect ShadowMove is also included with the Windows operating system. Log exports and searches can be done with the Windows Event Viewer; however, for convenience, a free trial of DataDog Cloud SIEM was used as the functionality and filtering capabilities of the SIEM far exceed those of the Windows Event Viewer. Since this is a research project intent on discovering vulnerabilities in a malicious piece of code, utilizing an API hooking tool like API Monitor was highly beneficial to take a closer look at API calls for research purposes. However, in a production environment, an API hooking tool is not strictly necessary for detecting ShadowMove.

Data Presentation and Analysis

To simulate a more realistic breach scenario, the virtual lab machine was left running throughout the day with light web browsing and other tasks being completed on it to generate logs. During this time, the TCPEchoClient.exe, TCPEchoServer.exe, and PoC.exe commands were executed, and the socket was duplicated. During the attack, the APIs from PoC.exe were hooked to prove that the socket duplication successfully occurred and to demonstrate that all DLLs and library files were being called successfully. Windows 10 forwarded all Windows Security logs to DataDog Cloud SIEM during this timeframe. A four-hour timeframe within which the attack occurred was selected to investigate as this would be a realistic window within which an analyst may need to search for the execution of potentially malicious software. This section will explain why analysts must find meaningful methods for narrowing the data collected from Windows systems to pertinent timeframes and log types. Following will be a presentation of the data, an explanation of the logs collected over four hours, API calls of interest from the API Monitor software showing how ShadowMove successfully executed and duplicated a handle, and the pertinent data gathered from the Windows Security logs and how that data was filtered out of the other four hours of data.

Data Presentation

Log File Size Reduction for Manual Inspection Simplification

As explained in Section III, collecting logs and forwarding them to a SIEM solution is straightforward; however, the number of logs generated by a sole source can be astronomical. The logs were generated on a mostly idle virtual machine running very few executables or services. If the time of the malicious executable execution is known, narrowing the timeframe to a shorter period or filtering out logs that are unneeded for detection is vital for detection purposes. Most enterprise systems will have hundreds or thousands of endpoints generating more logs than the virtual machine used for testing, so knowing the indicators of compromise is vital in detecting a ShadowMove. Figure 2 demonstrates this by displaying a four-hour period within which over 38,584 Security events were logged and sent to the SIEM tool.

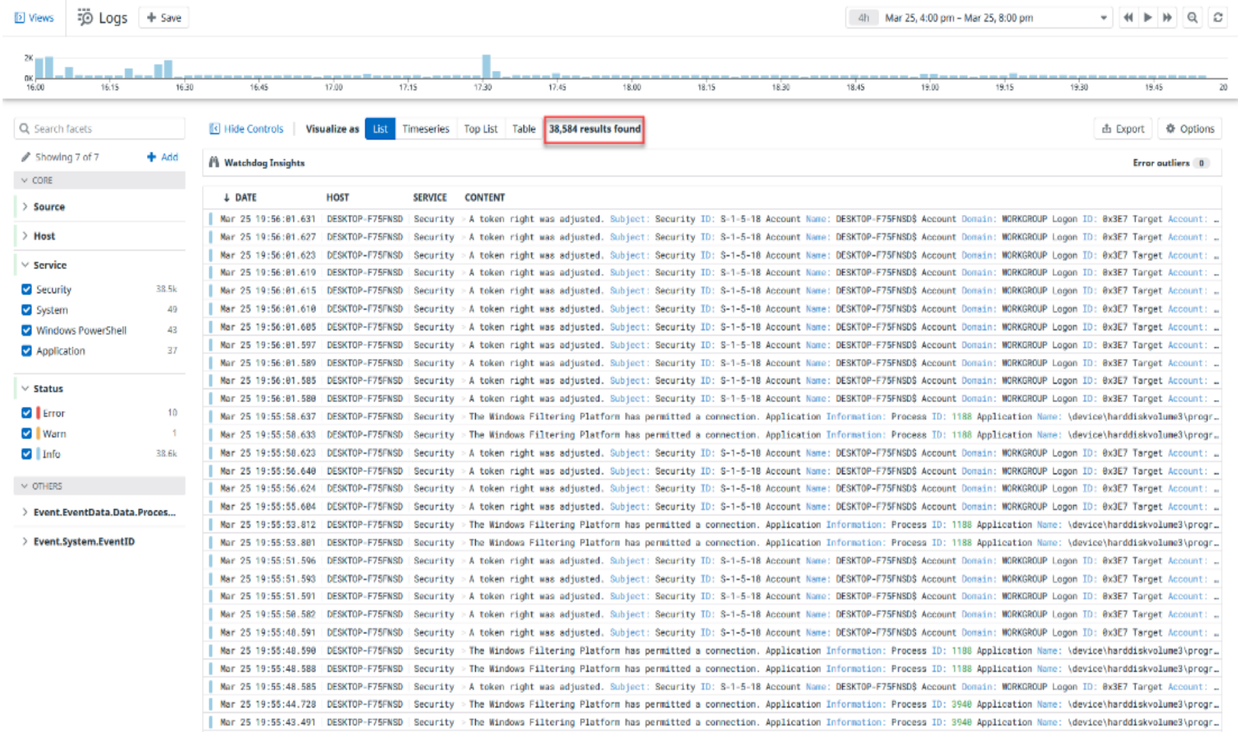

Figure 2. A screenshot from DataDog SIEM displaying the number of logs generated in a mostly idle four-hour period. Before applying any filters there were 38,554 log files to parse.

To narrow the scope of logs, filters were implemented on the DataDog SIEM only to include the Windows Security Event IDs related to ShadowMove. These event IDs are 4663 "An attempt was made to access an object", 4688 "A new process was created", 4689 "A process has exited", and 4690 "An attempt was made to duplicate a handle to an object". This filter is displayed in Figure 3. Furthermore, once the filter was applied, the number of events listed was reduced to 3,238 as demonstrated in Figure 4. Once the filter was applied to the target data, a comma-separated values file was exported and downloaded for filtering, searching, and manual inspection using Microsoft Excel.

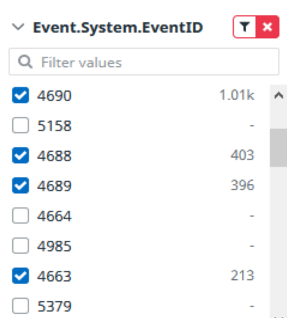

Figure 3. Screenshot taken from DataDog SIEM showing the filter settings applied to reduce the number of logs to review as well as the number of those logs generated in the four-hour period.

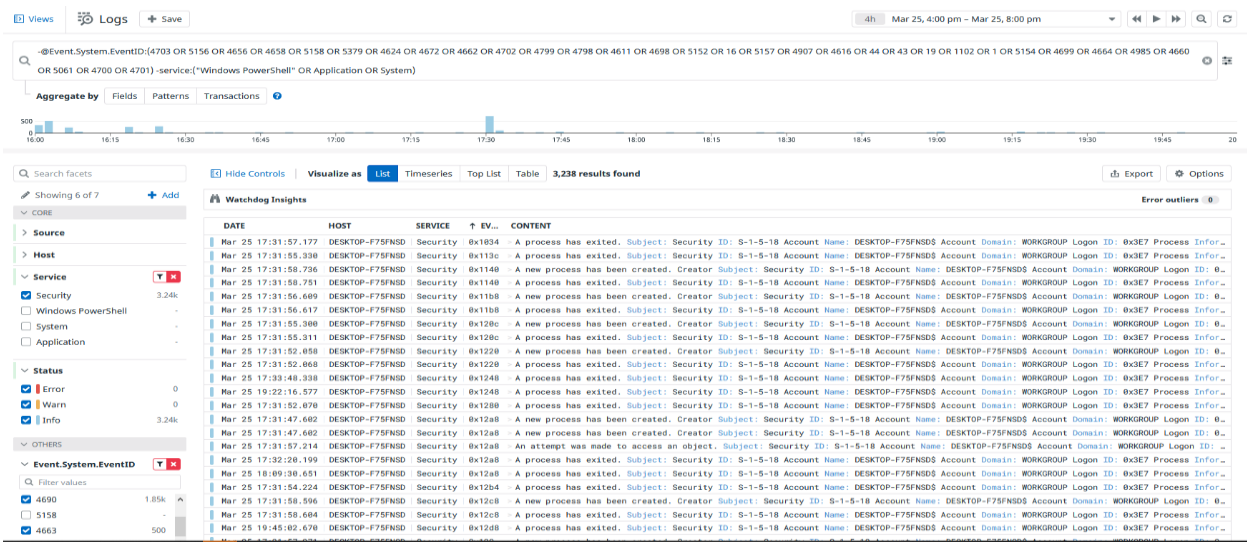

Figure 4. Screenshot of DataDog SIEM Log View with the ShadowMove Hunting filter applied. Notice the substantial reduction in log volume by applying a simple filter based on Event ID.

API Monitor and Static Analysis of API Calls

Manual inspection of the API Monitor output shows the ingenuity behind the ShadowMove attack. Each stage of the attack, as outlined in Section III, is caught during the execution of ShadowMove by API Monitor. This section contains screenshots of the API calls made during runtime. It shows that successful socket duplication occurs and demonstrates that the packaged libraries are one of Shadow Moves greatest strengths and its biggest weakness. Something to note in figures five through eight is that all API calls are pulled directly from PoC.exe; therefore, Log ID 4663 will not trigger on ntdll.dll nor WS2_32.dll when PoC.exe is executed. In Figure 5, the ShadowMove program (PoC.exe) is executing the first step of ShadowMove. Using the ntdll.dll packaged in the ntdll.lib file, PoC.exe is calling the API NtQuerySystemInformation to search for the AFD handle into which it can inject. Online 1028 the injectable handle is found as noted by "Return Value: STATUS_SUCCESS".

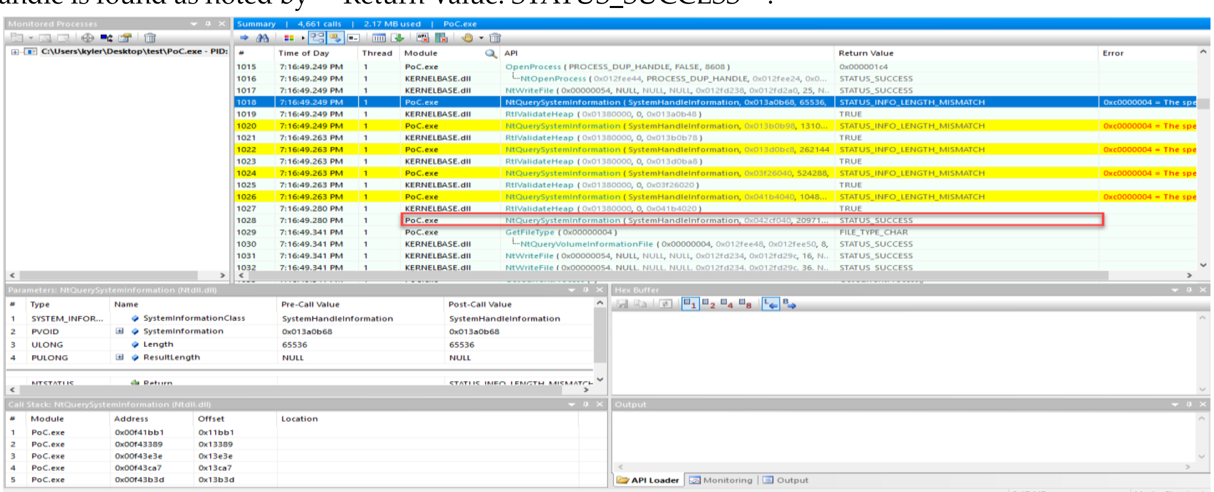

Figure 5. PoC.exe calls ntdll.dll from the linked library file to query system information to find an injectable AFD handle.

In Figure 6, PoC.exe uses ntdll.dll to attempt and create a new object handle by duplicating the object handle of the discovered AFD handle discovered in stage one. Again, note that the API call is originating from PoC.exe and not ntdll.dll—this is due to the linked library files.

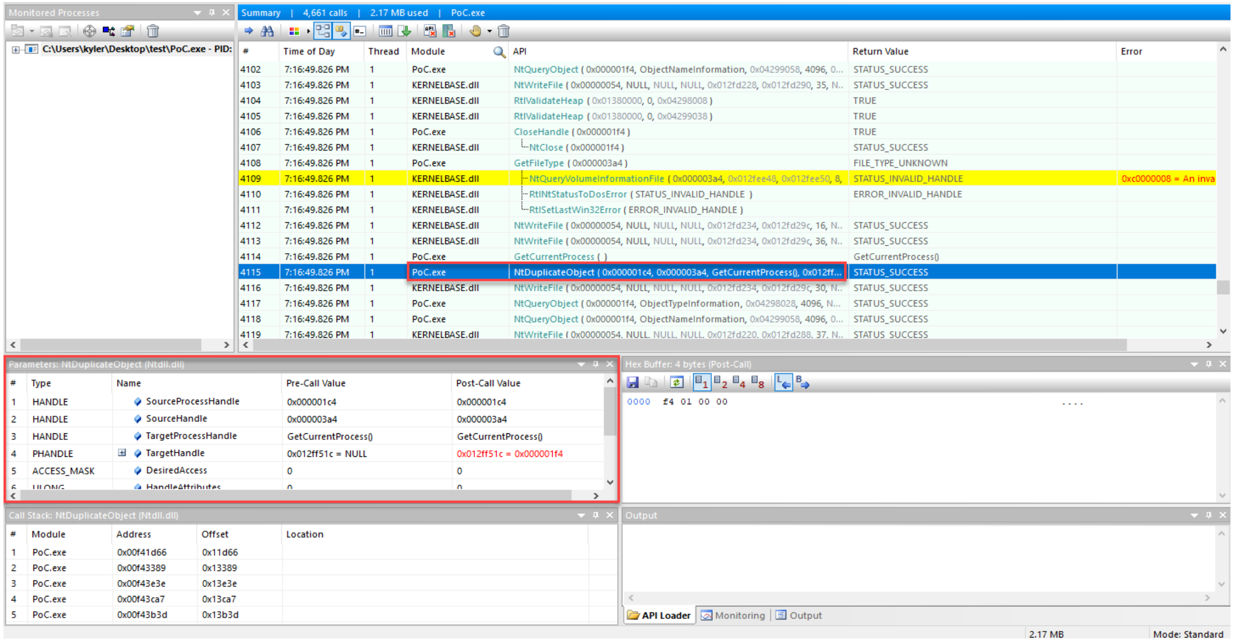

Figure 6. PoC.exe calling NTDuplicateObject to duplicate the AFD handle to use in a socket connection attempt.

Once ShadowMove successfully duplicates the object, PoC.exe calls WSADuplicateSocketW from WS2_32.dll to create the special protocol structure for the final stage of the attack as shown in Figure 7.

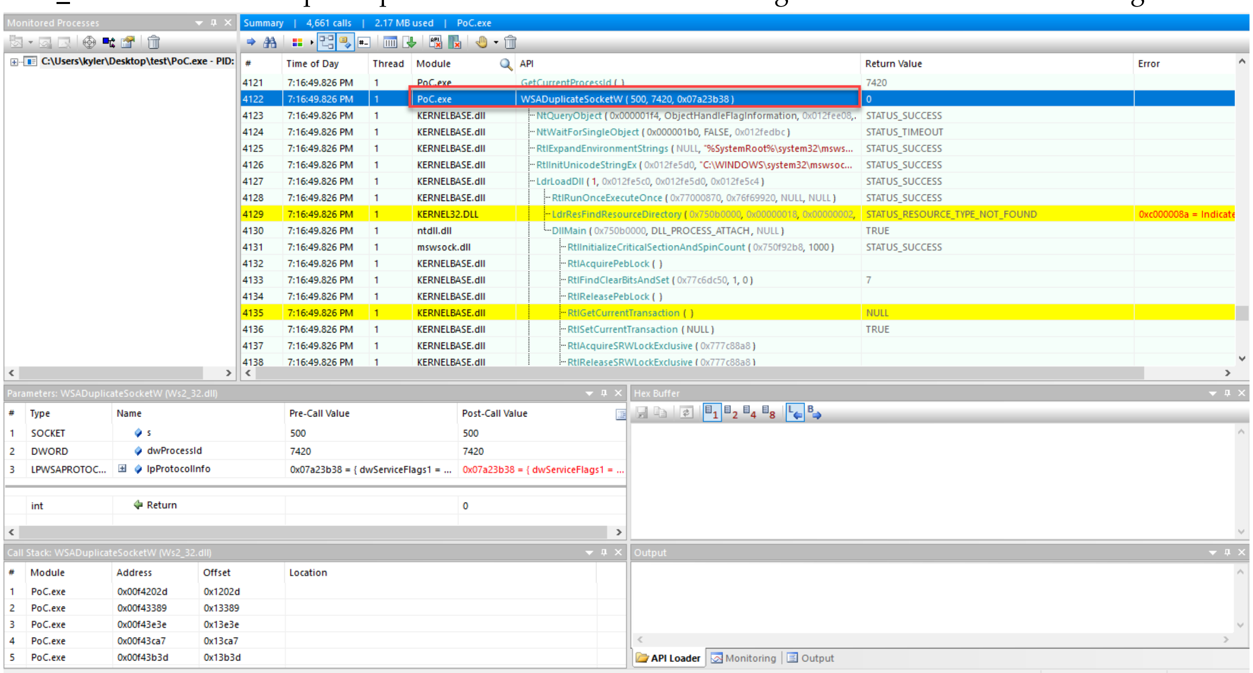

Figure 7. PoC.exe calls WSADuplicateSocketW to create the special protocol structure that will be used to connect to the socket in the final stage of ShadowMove.

Finally, once the special protocol structure is created in the third stage of the attack, PoC.exe calls WSASocketW from WS2_32.dll and provides the information WSADuplicateSocketW to connect to the duplicated socket. In Figure 8 the socket connection takes place on line 4547, and the data sent a request can be seen on line 4557, while the reception of data can be seen on line 4560.

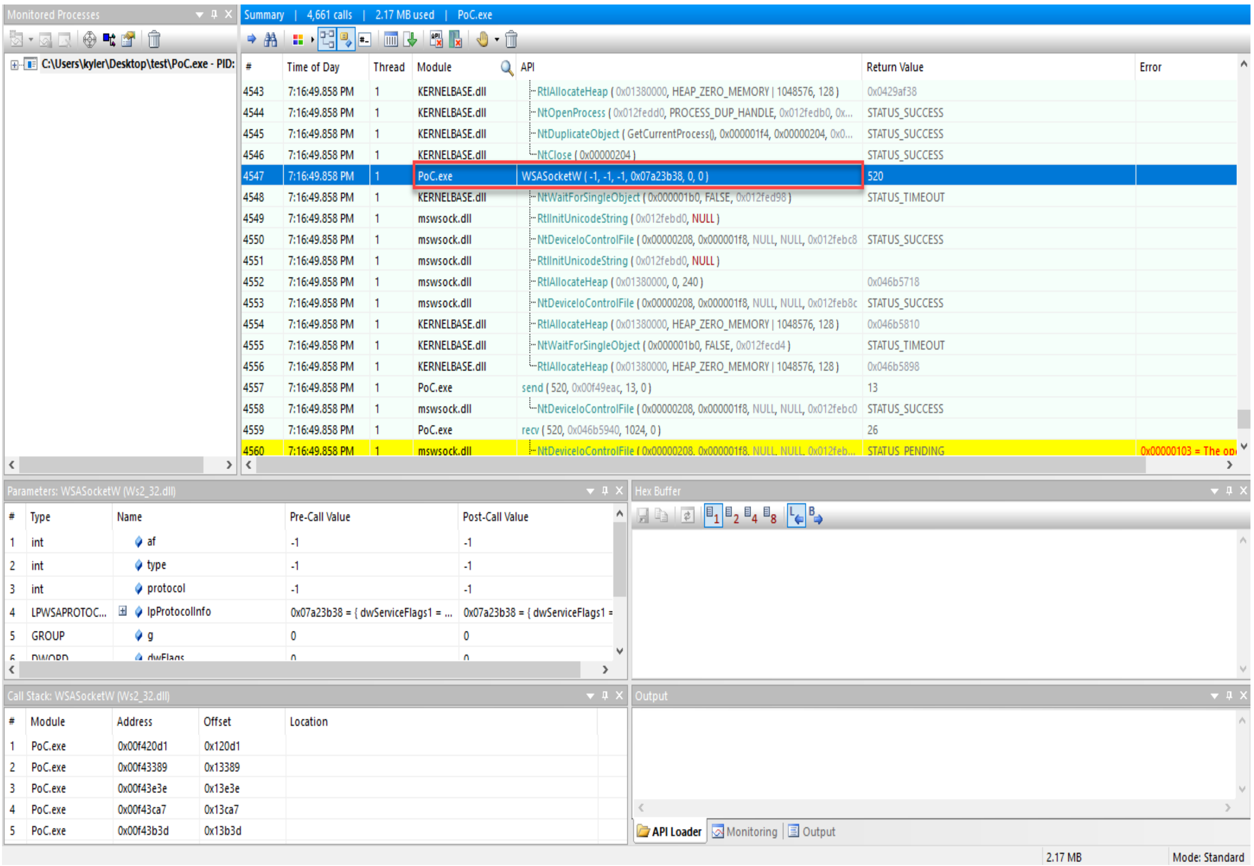

Figure 8. PoC.exe calls WSADuplicateSocketW to duplicate the socket and connect.

When an attempt to duplicate a handle is made, a Windows Security event 4690 is generated. These events are not rare, and within the four-hour window within which ShadowMove took place, there were over 1,850 handle duplication events logged as demonstrated in Figure 9. If a suspicious program name or location is duplicated, it may raise red flags for an analyst; however, finding the process name in the sea of logs is extremely difficult. Image nine displays the number of logs generated during the four-hour timeframe relevant to the investigation.

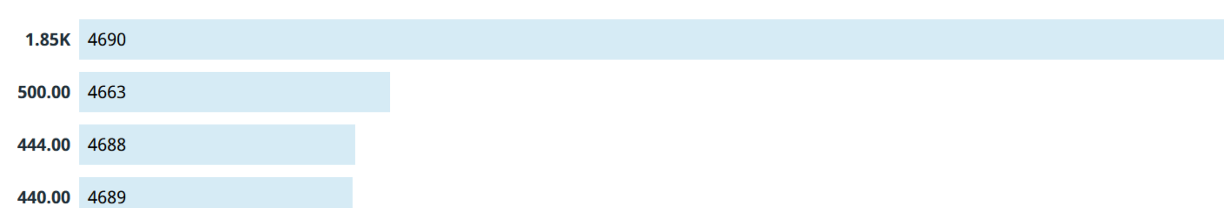

Figure 9. A large number of handle duplication events takes place every hour on a Windows system.

Finally, Figure 10 displays an important log generated during ShadowMove. This log is an auditing log for item access and is generated on C:\Windows\SysWOW64\wshqos.dll. This DLL is responsible for loading library files from executables. This event, in my research, rarely takes place and will be the key to finding an instance of ShadowMove taking place among the numerous log files generated by a system.

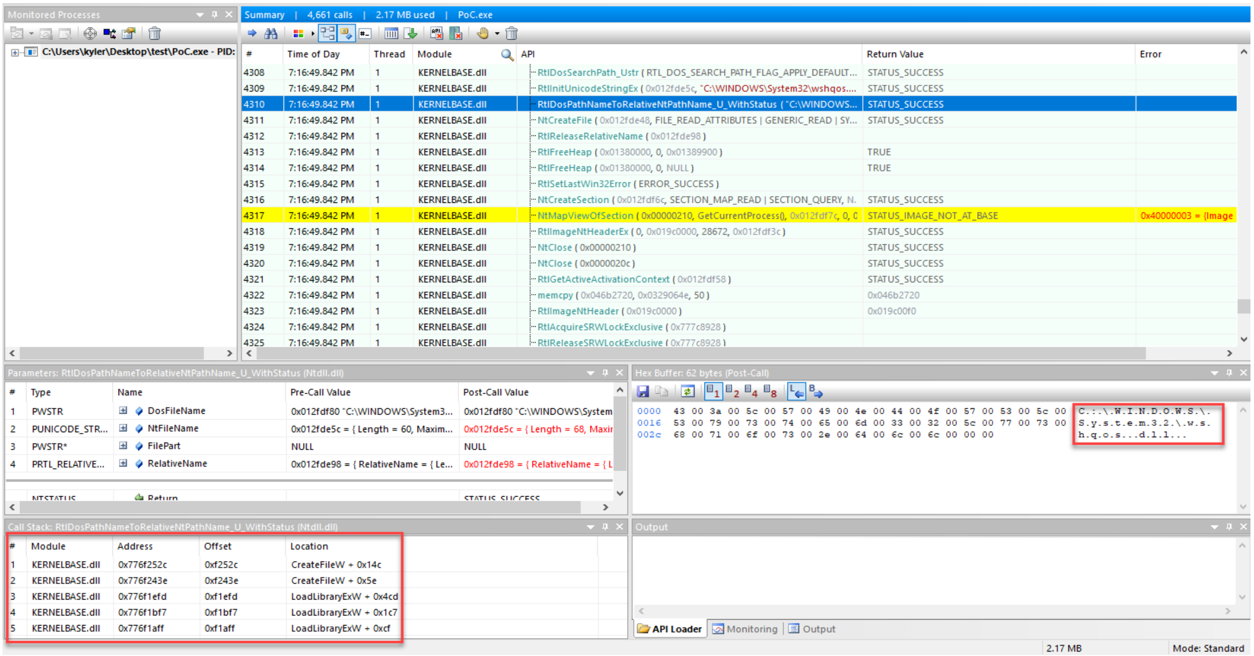

Figure 10. Access to wshqos.dll is made to load library files into the ShadowMove process. This is one of the few areas where ShadowMove directly interacts with the operating system.

Data Analysis

Because ShadowMove touches so little of the core operating system, handling duplication events are so common, and DLLs used during the attack are innately trusted by Windows, automated detection of the malicious software is difficult. In analyzing the data, we aim to propose a method by which analysts can narrow down whether ShadowMove may have occurred. Manual analysis of the API calls will always be necessary to prove beyond the shadow of a doubt that ShadowMove occurred; however, this methodology that we propose will allow the analyst to narrow down the list of suspect processes to a level where manual analysis is possible. The data analysis method is done in Microsoft Excel by manipulating the filter options on the csv downloaded from DataDog Cloud SIEM. Since wshqos.dll is the one file on the operating system that ShadowMove directly interacts with, the first filter is set on the message column, searching for any cells that contain the string wshqos.dll:results of this can be seen in Figure 11. Applying this filter significantly narrows the field as only eight cells contain the string wshqos.dll.

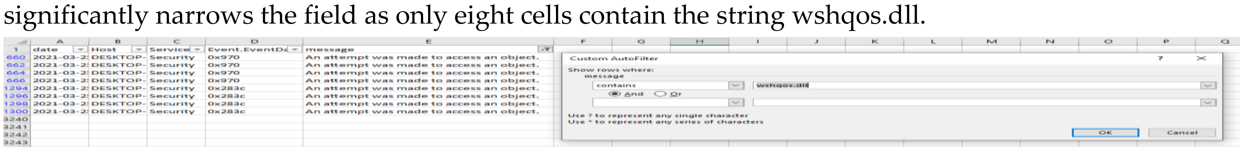

Figure 11. Setting the filter query to only list cells where the string wshqos.dll exists in the message column.

These log messages provide detailed information on the process name that is called the API from the wshqos.dll file. In both cases, the process name is PoC.exe, and the process id is either 0x970 or 0x283c. The second step to manually analyzing whether a ShadowMove occurred is to search the message column for the discovered process ids. Doing so returns 32 results and begins to build a process flow for executing both processes: the results of this search can be seen in Figure 12.. An analyst, at this point, would note that the process was created, attempted to duplicate objects and access objects being audited, and then exit. This follows the operating procedure of ShadowMove, and upon closer inspection, if the process name is unknown or running from a strange directory, an unwanted program is likely being executed within the environment.

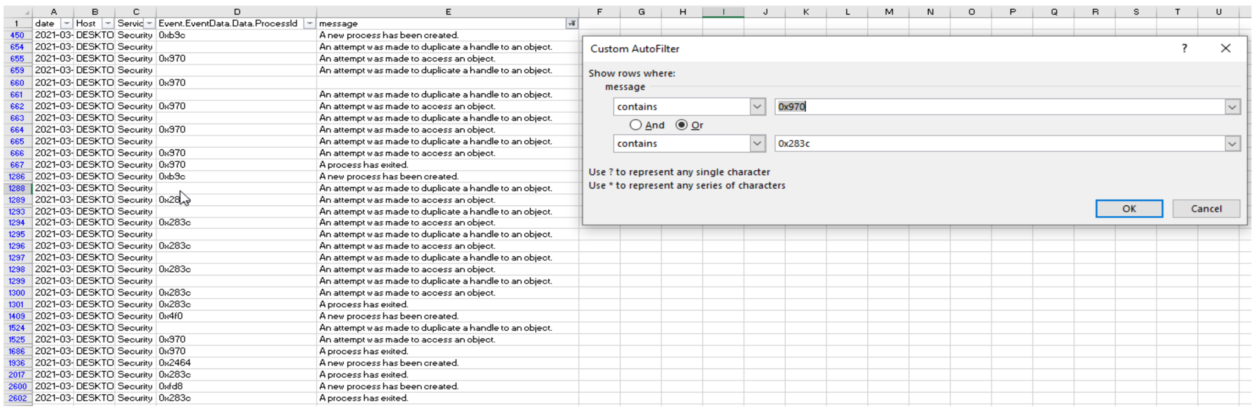

Figure 12. The ShadowMove Process ids are filtered and an analyst is able to view most steps of the ShadowMove process by filtering down Windows Event Logs.

Once the process name and location are discovered, an analyst should sandbox the unknown application and determine its purpose and whether it is malicious.

ShadowMove is a sophisticated piece of malware and, due to its programming, requires a high level of manual analysis to determine what it is doing. However, an analyst can use the data processing techniques outlined in this section to apply Windows auditing to specific DLLs, collect the pertinent log files, filter the logs, and determine whether a ShadowMove may have occurred. By filtering in this manner, an analyst could find file names and process ids to investigate further; however, without statically or dynamically analyzing the code, it would be impossible to determine with complete certainty that ShadowMove occurred.

Conclusions and Recommendations

This paper introduced traditional methods for lateral movement in Windows systems and well-known defenses for protecting systems from malicious lateral movement. Likewise, the paper explains why lateral movement is such a persistent issue and postulates that the most secure systems are non-networked systems which is the only surefire way to stop lateral movement. However, this solution will also significantly impede standard business practices. The authors also research new methods proposed for lateral movement detection, including graph-based and machine learning-based models. Finally, the paper presents ShadowMove, how it functions, and a new method for detecting ShadowMove, which has not been detectable to general user's knowledge. If an analyst is armed with the knowledge of socket duplication and how it can be used to duplicate network handles and inject anything into preexisting

TCP streams, the method that we propose will lead an analyst to an executable file for manual or dynamic code analysis. Therefore, it is confirmed that ShadowMove is a legitimate threat and is excellent at evading detection as it hardly touches the host operating system. Also, we were able to monitor API calls during runtime, confirm that socket duplication is feasible and possible without setting off many alerts, and determine a method for detecting ShadowMove as it touches the host operating system. Likewise, we developed a manual filtering process using nothing but Windows Auditing and Event Logs to find a process name and process ID that may be conducting a ShadowMove. While not the most elegant solution, it functions as intended and will detect ShadowMove if the analyst knows what to look for. The key to ensuring success in detecting a ShadowMove lies in auditing the correct files. In addition, administrators should ensure they are monitoring for program access to specific DLLs related to socket duplication, library loading, and network communications. While these may be noisy and generate numerous logs, if a ShadowMove is considered present in the environment, the log generation may be the lynchpin in a system that either detects this novel lateral movement or does not.

Future Works

Creating a custom alert based on the wshqos.dll file access and subsequent handle manipulation events generated by the same host process would be a method for automating some detection processes. This is a ruleset that we plan to implement in a SIEM solution in the future. If the DLL access closely mirrors the access outlined in this research, it is likely that some method of ShadowMove is being committed. Another area we did not focus on in this research is comparing the log generation with numerous other programs to determine how many false positives may exist in an enterprise-level system. Since our research environment was only a single virtual machine running extraordinarily little software, this method may generate more false positives than we anticipated. The authors would like to spend more time studying in more feature-rich environments to determine whether my method will function as intended or generate multiple false positives. One of the major drawbacks of detecting ShadowMove is that it requires the analyst to determine exactly which API calls were made from specific DLLs. Since the attack uses specific functions from specific DLLs, the attack has a unique signature. However, since the signature is also based on standard Windows protocols used daily, differentiating a malicious ShadowMove from benign processes can be extremely tedious and difficult. Once ShadowMove is suspected, an analyst should manually observe the file during runtime to determine whether socket duplication occurred using specific function calls from DLLs. Solutions to investigate API calls exist but are typically manual processes. As malware becomes more sophisticated and attackers increasingly are using built-in operating system functions to execute attacks and bypass traditional Antivirus solutions, we believe it will be important in the future to continuously monitor specific function calls from DLLs. Traditional SIEM tools can monitor logs, but a solution that could hook DLL calls at runtime and log API calls from those DLLs would speed up analysis of potentially malicious software that is currently not alerted on.

Furthermore, incorporating some version of deep-process analysis with a machine learning architecture like the one proposed by Holt et al. could lead to significantly more secure systems. While the research we undertook focuses on a specific attack developed by Niakanlahiji et al., the methods used to detect the malicious takeover and utilization of Windows DLLs are not exclusive to this specific attack. Shadow Move is a specific and functional instance of DLL code being packaged and hijacked for malicious purposes. Still, similar methodologies are likely being utilized by other malware. They could be used to execute ransomware and other malicious software undetected on Windows Systems by taking advantage of this Windows feature. Additional work should be conducted to detect the hijacking of DLLs through packaged code.

Furthermore, the Windows operating system provides an auditing package that is robust and able to collect all the necessary information to detect these types of attacks. However, the requisite logs necessary for detection are not enabled by default on the Windows operating system. Work should be completed to educate analysts on the methodologies utilized by programs like Shadow Move and how to best leverage the Windows auditing platform to collect the information necessary to make an informed decision. Without altering the default values in the Windows Auditing Platform, these attacks will go unnoticed and be unsearchable either in the auditing platform or in a SIEM solution. Finally, additional research should be completed on other malware variants and organizations to determine if DLL hijacking and manipulating existing TCP streams have occurred. While it is unknown whether these attacks have been successfully conducted in the production networks, the proof-of-concept code studied in this research proves that the attack vector is feasible for hijacking connections, moving laterally, and doing so undetected. Therefore, it is important to continue to study this method of DLL packaging, hijacking, and manipulation to determine its utility and functionality in other attack frameworks.

References

[1] MITRE, *https://attack.mitre.org/matrices/enterprise/windows/*, accessed: Nov,**2020.**

[2] S. Shelley, Phishlabs by HelpSystems, *https://info.phishlabs.com/blog/phishing-number-1-data-breaches-lessons-verizon/*, accessed: Nov,**2020.**

[3] Verizon, *https://www.verizon.com/business/en-gb/resources/reports/2020-data-breach-investigations-report.pdf*, accessed: Nov,**2020.**

[4] V. Lynch, hashedout by The SSL Store, *https://www.thesslstore.com/blog/2013-target-data-breach-settled*, accessed: Oct,**2020.**

[5] R. Weiner, The Washington Post, *https://www.washingtonpost.com/local/public-safety/hacker-linked-to-target-data-breach-gets-14-years-in-prison/2018/09/21/839fd6b0-bd17-11e8-b7d2-0773aa1e33da_story.html*, accessed: Nov,**2020.**

[6] B.A. Powell, The Epidemiology of Lateral Movement: Exposures and Countermeasures with Network Contagion Models., *Journal of Cyber Security Technology*, **2019**, 4(2), 1–39

[7] MITRE, *https://attack.mitre.org/tactics/TA0008/*, accessed: Nov,**2020.**

[8] D. Han, A. Li, D. Rugerio, L. Chen, S. Xu, 22 Sept. 2020, Microsoft Documentation, *https://docs.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library/*, accessed: Nov,**2020.**

[9] P. Pedemakar, EDUCBA, *https://www.educba.com/what-is-smb/*, accessed: Oct,**2020.**

[10] Q. Radich, D. Coulter, M. Satran, Microsoft Documentation, *https://docs.microsoft.com/en-us/windows/win32/termserv/remote-desktop-protocol/*, accessed: Nov,**2020.**

[11] D. Marshall, E. Graff, Microsoft Documentation, *https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest*, accessed: Mar,**2021.**

[12] A. Niakanlahiji, J. Wei, Md. R. Alam, Q. Wang, B.T. Chu, ShadowMove: A Stealthy Lateral Movement Strategy In *Proceedings of The 29th USENIX Security Symposium*, 12 Aug **2020.**

[13] CVE Details The ultimate security vulnerability datasource, *https://www.cvedetails.com/cve/CVE-2017-0143*, accessed: Mar,**2021.**

[14] Q. Liu, J. Stokes, R. Mead, T. Burrell, I. Hellen, J. Lambert, A. Marochko, W. Cui, Latte: Large-Scale Lateral Movement Detection In *Proceedings of IEEE Military Communications Conference (MILCOM) 2018 Track 3 - Cyber Security and Trusted Computing, Los Angeles, California, United States of America*, 29-31 Oct **2018.**

[15]R. Holt, S. Aubrey, A. DeVille, W. Haight, T. Gary, Deep Autoencoder Neural Networks for Detecting Lateral Movement in Computer Networks. In *Proceedings on the International Conference on Artificial Intelligence(ICAI), Xuzhou, China* , 22 Aug **2019.**

[16] H. Sidati, B. Saket, N. Memon, Detecting Malicious Logins in Enterprise Networks Using Visualization, In *Proceedings of IEEE Symposium on Visualization for Cyber Security (VizSec)*, 24 Oct **2016.**

[17] A. Bohara, M.A. Noureddine, A. Fawaz, W.H. Sanderse, An Unsupervised Multi-Detector Approach for Identifying Malicious Lateral Movement, *In Proceedings of IEEE 36th Symposium on Reliable Distributed Systems (SDRS), Hong Kong, Hong Kong,* 26-29 Sep **2017.**

[18] Y. Yu, J. Long, Z. Cai, Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders, *Security and Communication Networks*. **2017**, 2017, 1–10.

[19] H. Liu, B. Lang, Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders, *Applied Sciences.* **2019,** 9(20) 4396, 1-28.

[20] A.M. Chandrasekhar, K. Raghuveer, Intrusion Detection Technique by using K-means, Fuzzy Neural Network and SVM classifiers, *In Proceedings of 2013 International Conference on Computer Communication and Informatics (ICCCI),* 4-6 Jan **2013.**

[21] H. Chen, L. Jiang, Preprint, *arXiv preprint,* arXiv:1904.02426. DOI: 10.48550/arXiv.1904.02426, submitted: Apr, **2019.**

[22] P. Gogoi, D.K. Bhattacharyya, B. Borah, J.K. Kalita, MLH-IDS: A Multi-Level Hybrid Intrusion Detection Method, *The Computer Journal*. **2013,** 57(4), 602–623.

[23] A. Pektaş, E. Basaranoglu, Practical Approach for Securing Windows Environment: Attack Vectors and Countermeasures, *International Journal of Network Security & Its Applications (IJNSA).* **2017,** 9(6), 13-27.

[24] T. Bai, H. Bian, A.A. Daya, M.A. Salahuddin, N. Limam, R. Boutaba, A Machine Learning Approach for RDP-Based Lateral Movement Detection, *In Proceedings of 2019 IEEE 44th Conference on Local Computer Networks (LCN),* 14-17 Oct **2019.**

[25] M. Ussath, D. Jaeger, C. Feng, C. Meinel, Advanced Persistent Threats: Behind the Scenes, *In Proceedings of 2016 Annual Conference on Information Science and Systems (CISS)*, 16-18 Mar **2016.**

[26] G. Kaiafas, G. Varisteas, S. Lagraa, R. State, C.D. Nguyen, T. Ries, M. Ourdane, Detecting Malicious Authentication Events Trustfully, *In Proceedings of NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan,* 23-27 Apr **2018.**

[27] H. Siadati, N. Memon, Detecting Structurally Anomalous Logins within Enterprise Networks, *In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security – CCS '17, Dallas, Texas, United States of America,* 30 Oct – 03 Nov **2017.**

[28] S.P. Liew, S. Ikeda, Detecting Adversary Using Windows Digital Artifacts, *In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, California, United States of America,* 09-12 Dec **2019.**

[29] M. Fujimoto, W. Matsuda, T. Mitsunaga, Detecting Abuse of Domain Administrator Privilege Using Windows Event Logs, *In Proceedings of the 2018 IEEE Conference on Applications, Information and Network Security (AINS)*, 21-22 Nov **2018.**

[30] S. Abe, First Improving Security Together, *https://www.first.org/resources/papers/conf2016/FIRST-2016-105.pdf*, accessed: Oct **2020.**

[31] S. Tomonaga, JPCert/CC Eyes, *https://blogs.jpcert.or.jp/en/2016/01/windows-commands-abused-by-attackers.html*, accessed: Oct **2020.**

[32] J. Oh, First Improving Security Together, *https://www.first.org/resources/papers/conference2014/first_2014_-_oh-_junghoon_-_forensic_analysis_apt_lateral_movement_in_windows_environment.pptx*, accessed: Oct **2020.**

[33] I. Plotnik, T.A Be'ery, M. Dolinsky, O. Plotnik, G. Messerman, S. Krigsman, System (Microsoft Israel Research and Development 2002 Ltd), *US9729538B2,* **2017.**

[34] A. Dulkin, L. Lazarovitz, (CyberArk Software Ltd), *US20160330220A1*, **2018.**

[35] T. Hudek, C. McClister, Cymoki, Y. Suzuki, A. Viviano, K. Varadharajan, E. Graff, Komsorg, Prashantchahar, S. Resnick, iudezeGit-zz, udezeGit, Microsoft Documentation, *https://docs.microsoft.com/en-us/windows-hardware/drivers/download-the-wdk*. Accessed: Mar **2021.**

[36] API Monitor, *https://apimonitor.com*, accessed: Mar **2021.**

[37] Datadog Infrastructure and Application Monitoring, *https://docs.datadoghq.com/agent*, accessed: Mar 2021.

[38] D. Simpson, A. Lobo, H. Yoshioka, M.H. Avedon, Onur, J. Hall, A.M. Gorzelany, A. Bichsel, Microsoft Documentation, *https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/basic-audit-object-access*, accessed: Mar **2021.**

[39] D. Simpson, A. Lobo, Onur, J. Hall, A.M. Gorzelany, A. Bichsel, N. Schonning, Microsoft Documentation, *https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/basic-audit-process-tracking*, accessed: Mar **2021.**

[40] Datadog Infrastructure and Application Monitoring, *https://docs.datadoghq.com/integrations/active_directory/*, accessed: Mar **2021.**