5-2021

Lateral Movement in Windows Systems and Detecting the Undetected ShadowMove

Kyle Rozendaal
kyle.rozendaal@protonmail.ch

Lateral Movement in Windows Systems and Detecting the Undetected ShadowMove

by

Kyle Thomas Rozendaal

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

May, 2021

Abstract

Lateral Movement is a pervasive threat that exists 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 completed into preventing lateral movement in enterprise systems. However, attackers are using more 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 is conducted, outlining new 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 method for moving laterally is introduced and studied, and an original method for detecting this method of lateral movement is proposed.

Acknowledgements

A big thank you to Dr. Amirreza Niakanlahiji, Dr. Jinpeng Wei, Md Rabbi Alam, Qingyang Wang, and Bei-Tseng Chu for inspiring this research with their research into novel lateral movement techniques and for developing ShadowMove. Especially to Md Rabbi Alam and Jinpeng Wei for their patience and assistance sharing their proof-of-concept code and helping this C++ novice troubleshoot errors in execution.

A thank you to my wife and family for putting up with my long hours hiding in the basement completing this research.

Table of Contents

List of Figures

Chapter I: Introduction

Introduction

Lateral movement is a technique outlined in the MITRE ATT&CK framework and is a major problem in enterprise networks during cyber-attacks (Matrix - Enterprise | MITRE ATT&CK®, 2020). 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 to attempt to move closer to the target system containing the data they wish to manipulate for financial benefit.

Many researchers focus on earlier phases of the attack chain because once lateral movement has begun, the attackers have already breached the perimeter and it becomes extremely difficult to contain the damage. Phishing, however, plays a key role in explaining why detecting and preventing lateral movement is important. According to phishlabs.com and the 2019 DBIR by Verizon, Phishing attacks were a key component in 32% of all successful data breaches (Shelley, 2019; Verizon, 2020). Typically, when a phishing attack is successful, there is a loss in 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 (Matrix - Enterprise | MITRE ATT&CK®, 2020) if a high-level administrator reveals their username and password information. Likewise, even compromising the integrity of a standard user could bypass a 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 typically 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 the field of cyber-security and threat intelligence. Lateral movement has been difficult to detect in the past since the ability to move laterally between systems is a key component in networked Windows environments. Many technical users need access to multiple systems throughout the day and protocols like remote desktop protocol (RDP) and SSH are important business tools to help companies achieve their goals. However, the same tools and methods that 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 took place 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 point-of-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 customer’s credit card information which cost Target nearly $300 million (Lynch, 2017; Weiner, 2018).

Problem Statement

Lateral movement is a pervasive threat during cyber-attacks and often closely resembles legitimate traffic. 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.

Nature and Significance of the Problem

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 extremely 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 the sensitive information, the initial point-of-compromise will be any public-facing machine they can find. Once attackers gain access to a machine, 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 takes place in every large data breach, it is an important problem to study. Some researchers have created research papers outlining the nature of the problem and exposing the pervasiveness of the threat (Powell, 2019).

Lateral Movement is Preventable:

Lateral movement is entirely preventable. By creating a series of systems that have no interconnectivity by locking down firewall rules and disabling protocols, lateral movement attacks would be impossible to execute. However, preventing attackers from moving laterally also prevents legitimate users from moving laterally. Networked computers, networked systems, and a series of machines working in tandem with shared access makes a business more profitable and efficient. It is understood that a computer connected to the internet is vastly superior in its capabilities than a computer without an internet connection. Likewise, in business environments, a non-networked workstation is going to 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 the use of 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 themselves.

Lateral Movement is difficult to detect:

Attackers use numerous methods for moving laterally through networks (Lateral Movement, Tactic TA0008 - Enterprise | MITRE ATT&CK®, 2018). Many of these methods require the attackers to capture password hashes and user credentials to move onto the next system. When attackers move in this way, it simply appears as though 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. There are indicators of lateral movement that can 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 have designed their methodologies for detection around these slight differences.

Objective of the Study

The goal of the study is to supply a broad overview of lateral movement techniques, defenses. Online sources and libraries like MITRE have troves of information concerning the practical application of lateral movement prevention as well as the methods protected against. Second, a literature review will be provided covering proposed methods for improved detection of lateral movement in enterprise systems. Finally, this study will demonstrate a novel method proposed in one of the research papers, demonstrate how it functions to bypass detection, and create a detection method to detect this stealthy lateral movement—a feat which has not yet been completed.

Study Questions/Hypotheses

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?

Network Complexity

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

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 yet?

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 DLL is a library that contains code and data that can be used by more than one program at the same time (Deland-Han, 2020).
- **Server Message Block:** Server Message Block is a network communication transfer protocol to provide shared access to files, printers, ports between the networks (Pedemakar, 2020).
- **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 (Schofield et al., 2018).
- **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 (Marshall et al., 2018).
- **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 dynamic link libraries 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.

Summary

Lateral movement is a major 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 extremely difficult to detect as it typically appears as 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.

Chapter II: Background and Review of Literature

Introduction

The purpose of this literature review is to provide the reader with a deeper understanding of the work conducted in the field of lateral movement detection attempting to solve some of the major problems that exist in differentiating malicious traffic from regular traffic. This chapter will be broken up 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 research.

Background Related to the Problem

When approaching the problem of lateral movement, there are two main questions that 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 the background portion of this literature review focus on. 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 is covered as well as proposed defenses. A diagram outlining common tactics to prevent lateral movement is provided in Appendix A for engineers looking for easy solutions to cover the most attack methods. A review of literature proposing new types of detection for more precise differentiation and remediation will be provided following the MITRE review.

Literature Related to the Problem

MITRE ATT&CK Framework TA008 -- Lateral Movement

MITRE outlines numerous attack methodologies for lateral movement in their ATT&CK framework. In this section all methods described and their recommended mitigation methods for fixing the vulnerabilities will be discussed. The purpose for this section is to highlight common methodologies employed by attackers when moving through a Windows environment. The methodologies covered by MITRE are common exploits with well-known mitigations. In writing this section, this research aims to highlight the fact 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 use the exploitation of 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 that was 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 found inside Windows Server Message Block (SMB) and allows for a remote attacker with no credentials to gain SYSTEM level privileges on the target machine. (“CVE-2017-0143: The SMBv1 Server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and R2 SP1; Windows 7 SP1; Windows 8.1; Windows”, 2017).

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 discovered, 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 Spearphishing. Internal spearphishing is when an attacker compromises an internal email address and uses it to gain the trust of other internal users in order to trick them into sharing passwords or other sensitive data. 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 spearphishing attacks is extremely difficult as an initial breach has already occurred, and all attack traffic looks like standard email traffic. Employee awareness programs and employee training will help reveal internal spearphishing 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, open SMB file servers, network drives, or removeable media. By transferring attack tools to other systems, attackers can connect to and create a backdoor on whatever system it they place it on giving them deeper persistence in the network.

Some of the common mitigations for this type of attack include filtering network traffic to ensure only known devices and addresses are communicating with secure channels like SMB or SSH. Another method for preventing lateral tool transfer is to implement 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 have the capability to 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 a new session is not created by the malicious actor. 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 that 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 to use 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 airgaps, or to increase the likelihood of reaching difficult-to-reach machines, attackers may copy malware to removeable 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 automatically execute malware when a user inserts the removable media device into a new computer. Likewise, limiting the use of 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 gaining access to 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 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 are able to 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 remotely install applications on systems, so mitigating a threat like this can be difficult. Some companies will also have custom software that they may want to push, and it may be unsigned. Companies should ensure, that if they are going to use a remote deployment tool, that the 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 such that the intended action of the user is still executed so as not to raise suspicion. However, the malicious script will run and allow for deeper access to the network.

Mitigating shared content tainting 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 gain access to alternative authentication materials like Kerberos Tickets, Application Access Tokens, Authentication Tickets, or Web Session Cookies to bypass the password requirement to access the service. 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 include 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 present on the network.

Latte: Large-Scale Lateral Movement Detection

This research team discusses 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 and discovering an infected computer. “Latte analyzes large-scale event logs collected from operational networks” (Liu et al., 2018). Their system analyzes 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 this team 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 the path an attacker took 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 of 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 (Liu et al., 2018). 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 the malicious paths taken between workstations were discovered by analysts in the eleven top results, the researchers determined their forensic analysis module to be a success.

The authors admit 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-day period over 44 Million connections were tied as the most suspicious to generate fewer false positives, the authors recommend 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, does propose a method to how analysts can differentiate between malicious and non-malicious traffic. By correlating Kerberos Service Tickets with remote execution and analyzing rare paths using a map of the network, 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.

Deep Autoencoder Neural Networks for Detecting Lateral Movement in Computer Networks

This research team researched the use of Deep Autoencoder Neural Networks in detecting lateral movement in networked computers. They begin by outlining the fact that many other researchers have researched using neural networks to aid in detecting intrusions in computer networks. However, this team differentiate their research from past research endeavors by setting out to solve the problem of lateral movement rather than general intrusion detection.

The team, led by R. Holt, used the Los Alamos National Laboratory dataset to train and test their neural network. The Los Alamos National Laboratory dataset covers a period of 58 days and is over 73 gigabytes in size. Therefore, the team 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 (Holt et al., 2019). The developmental dataset included all the red team data from the Los Alamos Dataset as well as all normal traffic from the computers compromised by the red team. Researchers added a random sampling of data to make the developmental set more varied and to avoid overfitting of the data. Researchers created the test dataset in the same manner with the addition of 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 a model designed 6-5-4-3-2-3-6 (Holt et al., 2019). 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 the team performed worse than the semi-supervised model they reference in their related works section. However, the semi supervised model proposed by another research group covered in this paper (Sidati et al., 2016), requires a human analyst to aid in the detection of anomalies and is not fully automatic like the model proposed by R. Holt and his team. Furthermore, the model proposed by Holt and his team was more accurate than the model proposed by Bohara and their research group (Bohara et al., 2017). While the results show positive progress towards the goal of automating intrusion detection and lateral movement detection using autoencoders, further research must be made 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 have 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 (Liu et al., 2018; Chandrasekhar & Raghuveer, 2013; Chen & Jiang, 2019; Yu et al., 2017; Liu & Lang, 2019). Many semi-supervised models perform extremely well when pairing the judgement of a human with the pattern recognition of a computer (Gogoi et al., 2013). Methods researched by teams like Holt’s team show 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.

Practical Approach for Securing Windows Environment: Attack Vectors and Countermeasures

Abdurrahman Pektaş and Ertugrul Basaranoglu introduce a new method for conducting penetration tests within a Windows Environment. They make the claim 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 (Pektaş & Basaranoglu, 2017).

The authors begin their article by outlining the basics behind other penetration testing methodologies introduced by companies like OWASP and the CE-Council but quickly begin work on demonstrating why their Microsoft Domain Environment Penetration Test Methodology (MSDEPTM) is superior for testing Windows environments. The authors introduce a ten-step systematic process for attacking Windows environments and explain 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 successfully breach a Windows environment. The authors break down their methodologies in the ten-step penetration test method.

Section four covers mitigation techniques for preventing unauthorized access of systems as laid out within section three. While comprehensive in scope, the amount of detail in preventing certain methodologies is lacking. While this is a paper that 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 outlines that since they provide more steps, specific tools for attack and mitigation, as well as offering different techniques that their method competes with other attack methodologies for conducing 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 the purposes of this starred paper, this resource is helpful in outlining novel methods for exploiting Windows environments for lateral movement as well as potential measures to prevent against lateral movement. Many of these activities are also outlined in the MITRE ATT&CK Framework and will be covered in future sections. This research is helpful in developing a broader understanding of tools and techniques available to network defenders and how malicious lateral movement may be defended against.

A Machine Learning Approach for RDP-based Lateral Movement Detection

Some researchers propose a new method for classifying remote desktop protocol (RDP) sessions in Windows environments. Using datasets from the Los Alamos National Laboratory and supervised machine learning algorithms, the authors propose a new method for detecting and sorting through RDP sessions to better classify malicious lateral movement within a Windows environment. The research team concludes their research by comparing their developed method to state-of-the-art methods and gauge their effectiveness based off another model's performance (Bai et al., 2019).

The authors begin their research with a literature review of other authors that have tried to classify malicious RDP sessions using machine learning algorithms the Los Alamos National Laboratory Dataset (LANL). The team critiques the method proposed by the team led by Martin Ussath (Ussath et al., 2016) for being unwieldy in production environments, although the learning algorithm was efficient at detecting malicious RDP sessions. Furthermore, the authors critique Kaiafas' team (Kaiafas et al., 2018) 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.

The team levels 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 compared to normal activities is extremely small. Furthermore, the red team activities are launched from four different machines and take place during specific timeframes. For this reason, they conclude 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’s rather than generalized patterns in the malicious RDP activities (Bai et al., 2019). To solve this problem of overfitting the training data to specific activities generated by specific machines at specific time intervals, the research team proposes 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 (Bai et al., 2019).

Using their new combined dataset, they 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” (Bai et al., 2019). The authors then compare their model to another top performing model proposed by Kaiafas et al (Kaiafas et al., 2018). Using their dataset, the researchers were able to reduce the number of inputs and abstract the data more completely than Kaiafas’s team and were able to return higher detection rates. In doing so, the team posits that their model is more useful in a production environment as it requires less data to run and is as effective as the more complex model (Bai et al., 2019). This model is useful in highlighting 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. Oftentimes, Windows Event Logs are overlooked as being clunky or not verbose enough. This research proves that Windows Event Logs can be utilized effectively for intrusion detection purposes when the correct filters are applied, and careful logic is utilized. The machine learning algorithm proposed by the team demonstrates novel methods for detecting and preventing lateral movement using common tools accessible to most security analysts and engineers.

Detecting Structurally Anomalous Logins Within Enterprise Networks

Hossein Siadati and Nasir Memon introduce 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 employ an anomaly detection system to detect out-of-character logins for users within the network (Siadati & Memon, 2017).

Siadati and Memon focus on credential based lateral movement during which the attackers steal valid user credentials through tactics like pass the hash and authenticate as valid users. These types of attacks are some of the most difficult to detect because they so 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 employ 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 thought to be 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 (Siadati & Memon, 2017). 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. The reason for this was that 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 the idea of monitoring standard behavior for users and flagging anomalous logins is a good theory, in practice, more information needs to 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 in narrowing the scope and improving the overall accuracy of the system. While some sign-in based anomaly detection system could be helpful in detecting novel lateral movement techniques, further studies into this subject will need to be done before this type of detection can be relied on solely for malicious lateral movement detection.

While not the most effective solution, the method of 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 taking place within the infrastructure.

Detecting Malicious Authentication Events Trustfully

The research team led by Kaiafas aim to solve the problem with anomaly detection outlined in the paper by Siadati and Memon: false-positive detections. The team tried to solve this issue by providing more contextual data surrounding the authentication to the classifiers (Kaiafas et al., 2018). By including more contextual data, they aim to reduce the number of false positives by classifying more accurately what normal behavior looks like.

The team 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 (Kaiafas et al., 2018)—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” (Kaiafas et al., 2018). 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” (Kaiafas et al., 2018). The authors use 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” (Kaiafas et al., 2018). In their experimentation the team 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” (Kaiafas et al., 2018). 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 also expected spread of user activity meaning the more popular users should be expected to authenticate more frequently during a given period of time.

After running the dataset through these classifiers, the authors fed the data to four different “ensemble learning techniques” (Kaiafas et al., 2018) for final classification. These ensemble learning techniques use multiple machine learning algorithms to classify and sort data. The ensembles they used were LogitBoost, Random Forest, Logistic Regression, and

Majority Voting. After training their systems with a subset of data from the Los Alamos National Laboratory Dataset, the research team measured the success of their systems by computing the false positive rate, false negative rate, balanced accuracy which is “the arithmetic mean of True Positive Rate and True Negative Rate” (Kaiafas et al., 2018), 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% of the data. 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. The team prove that their sorting methods are effective at reducing noise generated for the administrator.

This research is fundamental in feature choice for reducing the noise generated by network logs. The researchers supply many features that seem to truly 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.

Advanced Persistent Threats: Behind the Scenes

The research team led by Martin Ussath investigated 22 different APT attacks to gather the best practices used by many of the APT’s to attack networks. In doing so, Ussath and the team proposed to highlight better detection methods for commonly used attack structures (Ussath et al., 2016). To simplify the complexity of APT attacks, the researchers view three main categorizations of activities taken during an APT campaign: initial compromise, lateral movement, and command and control (Ussath et al., 2016).

After 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 preinstalled Windows tools like remote desktop protocol, windows management instrumentation, PowerShell, and PS Exec (Ussath et al., 2016).

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 the researchers is to exploit known vulnerabilities to elevate privileges. The authors propose that attackers exploited vulnerabilities because access to passwords and password hashes required administrative credentials (Ussath et al., 2016).

To detect malicious lateral movement, the team proposes 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 (Ussath et al., 2016).

Viewing the chart of 22 different APT groups created by the researchers gives 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 the team is provided in Appendix A and outlines the most common methods used by APT groups and gives a good overview of attacks to focus on defending against.

Literature Related to the Methodology

ShadowMove: A Stealthy Lateral Movement Strategy

The research team led by A. Niakanlahiji proposes a novel lateral movement strategy that takes advantages of built-in Windows features to jump between systems using existing connections while bypassing all modern AV detection (Niakanlahiji et al., 2020). The system works by duplicating socket connections and hijacking established FTP, TDS, and WinRM connections.

The system proposed by uses 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 recording when a certain TCP port is being 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 essential information the attacker may want to know when engaging in lateral movement as part of a cyber-attack.

The Socket duplicator duplicates sockets. On a windows system this is done by using 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 many modules responsible for crafting and reading from packets midstream and is responsible for crafting packets that can hijack FTP, MS SQL, and WinRM connection streams.

This team created a stealthy lateral movement technique that bypassed all traditional antivirus, endpoint detection and response tools, and IDS/IPS tools that were leveraged against the ShadowMove software. The authors do, however, 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 proves to be effective in bypassing 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 chapters III, IV, and V as the goal of this research is to invent a novel method for detecting this ShadowMove attack.

Detecting Adversary using Windows Digital Artifacts

In this paper, the Seng Pei Liew and Satoshi Ikeda propose a method for detecting advanced persistent threat adversaries in a Windows environment using nothing but native Windows artifacts (Pei Liew & Ikeda, 2019). The authors begin by outlining two key issues with detecting persistent adversaries in a Windows environment. The first issue is that attackers use benign file names or files to conduct their attacks to prevent signature detection and the second is that there are disparate configurations within Windows environments and the lack of conformity to a standard makes tracing paths difficult. To overcome these issues the authors, propose a machine learning based approach that observes digital artifacts left in all Windows systems. To do this, the authors also propose a new algorithm to learn the execution time of a process from the shipmate (Pei Liew & Ikeda, 2019). Using the data gathered from the Shimcache and the output of the machine learning algorithm, the authors propose an adversary detection system that, given a period of time, will 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 are 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 that are created by running the tools in the Master File Table, Shimcache, Prefetch, and Windows Event Log during execution. The authors explain their methodology for tracing an attack using these event artifacts and outline 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 (Pei Liew & Ikeda, 2019).

After explaining the details of the timing algorithm, the authors explain how their machine-learning based scoring algorithm can 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 timeframe accordingly. The scoring module takes a list of commands commonly used by attackers to execute distinct phases of an advanced persistent threat attack as outlined above (Pei Liew & Ikeda, 2019). The algorithm used for training is a Random Forest algorithm which is a black-box method of training. This means that the researchers know the data they put in, but the computations that take place 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% (Pei Liew & Ikeda, 2019). The results are not fantastic, and researchers were upset that certain 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 are only focusing 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 a narrow slice of time and decide on malicious behaviors that took place during a distinct amount of time. 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 chapters III, IV, and V as this research attempts to detect ShadowMove.

Detecting Abuse of Domain Administrator privilege using Windows Event Log

Fujimoto's research team set out to compare 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 (Fujimoto et al., 2018).

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 (Abe, 2016). The researchers include research done by Shusei Tomonaga at JPCERT/CC into common commands executed by attackers during an APT campaign (Tomonaga, 2016). This team, however, focused solely on correlating Windows Event Log 4688—A New System Process Has Been Created—to detect abuse of domain administrator privileges. They use research conducted by Junghoon Oh at

AhnLab to detect APT lateral movement using administrative shares to spread access (Oh, 2016.). The researchers also 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, the team includes research done by Idan Plotnik and Andrey Dulkin to detect golden ticket creation by logging Kerberos Service Ticket requests that have no prior Ticket-Granting-Ticket (TGT) associated with them (Plotnik et al., 2017; Dulkin et al., 2017).

This research team takes all these known methodologies for detecting abuse of domain administrator accounts and develop their own method with a high detection rate. Their method focuses on watching the domain controller for the creation of golden tickets or credential theft and does not detect abuse of all machines in the domain. This method, therefore, is not usable 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.

What Fujimoto and the rest of the team proposes is 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 categories of detection. Furthermore, not detecting abuse of a domain administrator account typically means that the attackers have compromised the entire domain and quick and exact remediation needs to take place at once. While novel in its scope, and thorough in its investigation of methodologies used to compromise domain administrator accounts, different methodologies must be used to detect and prevent against privilege escalation attacks in a more exact manner. The important artifact to take away from this research is that only Windows event logs were used to detect malicious movement in a system. It is possible to garner valuable information from intrinsic logging sources.

Summary

Whether proposing new methods for detection or presenting methods for preventing known attacks like the MITRE Group, 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 and 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 concluding section of the literature review was designed to give the reader a sturdy base of knowledge from which to draw when reading about the original methodology and tactics utilized in this research to detect ShadowMove, a feat which has not yet been completed.

Chapter III: Methodology

Introduction

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 toolsets 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.

Design of the Study

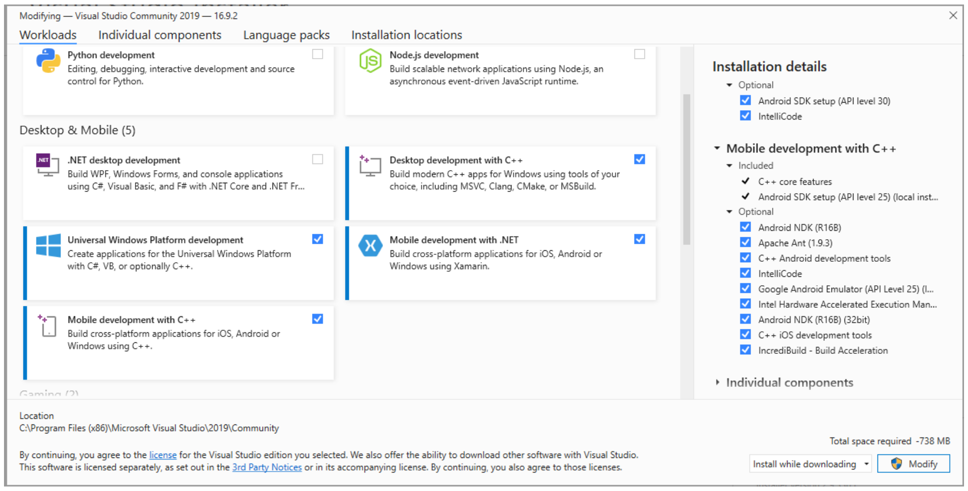
Virtual Machine Setup

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 in the virtual machine and updated to version 10.0.18363 with all required and recommended patches applied. 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.

Figure 1

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


Modifying — Visual Studio Community 2019 — 16.9.2

Workloads | Individual components | Language packs | Installation locations

Python development [ ]
Editing, debugging, interactive development and source control for Python.

Node.js development [ ]
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

.NET desktop development [ ]
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET Fr...

Desktop development with C++ [✓]
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Universal Windows Platform development [✓]
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Mobile development with .NET [✓]
Build cross-platform applications for iOS, Android or Windows using Xamarin.

Mobile development with C++ [✓]
Build cross-platform applications for iOS, Android or Windows using C++.

Gaming (2)

Installation details
- Optional
  - [✓] Android SDK setup (API level 30)
  - [✓] IntelliCode
- Mobile development with C++
  - Included
    - ✓ C++ core features
    - ✓ Android SDK setup (API level 25) (local inst...
  - Optional
    - [✓] Android NDK (R16B)
    - [✓] Apache Ant (1.9.3)
    - [✓] C++ Android development tools
    - [✓] IntelliCode
    - [✓] Google Android Emulator (API Level 25) (l...
    - [✓] Intel Hardware Accelerated Execution Man...
    - [✓] Android NDK (R16B) (32bit)
    - [✓] C++ iOS development tools
    - [✓] IncrediBuild - Build Acceleration
- Individual components

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required -738 MB

Finally, the Windows Driver Development Kit (WDK) (Hudek et al., 2020) to access the ntdll.lib and Ws32_32.Lib files for the ShadowMove build.

Software used for monitoring, diagnosis, and alerting. To monitor the software and Dynamic Link Library (DLL) function calls at runtime the software API Monitor v2 was utilized (“API Monitor - Spy and Display Win32 API Calls Made by Applications”, 2013). 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 (Datadog, 2019).

ShadowMove Essentials and Socket Duplication

A group of faculty and students from the University of Illinois Springfield, University of North Carolina at Charlotte, and Louisiana State University developed an attack methodology that utilizes Windows Dynamic Link Libraries 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 with 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 looks for custom executable code that either has patterns that are known to be malicious or has 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, by using trusted Windows dynamic link libraries, it ensures that the attacker will be able to 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 session using legitimate credentials, or the dumping/stealing of password hashes from Kerberos or LSASS. Once an initial foothold is gained on the system by the attacker, no credentials need be stolen from any location, nor do any new authenticated sessions need 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 makes it extremely difficult for an analyst to discover the lateral movement when following traditional attack patterns.

To understand how ShadowMove hijacks a typical socket duplication event in the Windows operating system, we must first understand how a typical socket duplication event is established by the system 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 uses 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, alternatively, 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 that was 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 is wherein ShadowMove has the ability to inject any data between the host and the guest without ever 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 (WDDK) (Hudek et al., 2020). The fact that the dynamic link library 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 I was 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 dynamic link libraries 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 dynamic link libraries like WS2_32 or ntdll. However, the DLL files include numerous functions that the Windows operating system needs to function. 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, I 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 some 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: first open the Local Security Policy application by search

“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 types of items (Simpson et al., 2017a). 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 (Simpson et al., 2017b).

ShadowMove utilizes ntdll.dll and WS2_32.dll and 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 labeled “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, as well as the process that called the file. If ShadowMove is utilizing the DLLs packaged with the Windows operating system, then 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 it 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 better hide its execution and intentions, 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 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 document 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. (Datadog, 2021a; Datadog, 2021b).

Summary

Just as most of the application code required to run ShadowMove is built into the Windows operating system, all the software required 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 extremely beneficial to take a closer look at API calls for research purposes, however, in a production environment, an API hooking tool is not necessary for detecting ShadowMove.

Chapter IV: Data Presentation and Analysis

Introduction

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 and 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 begin with an explanation as to 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 a four-hour period, 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.

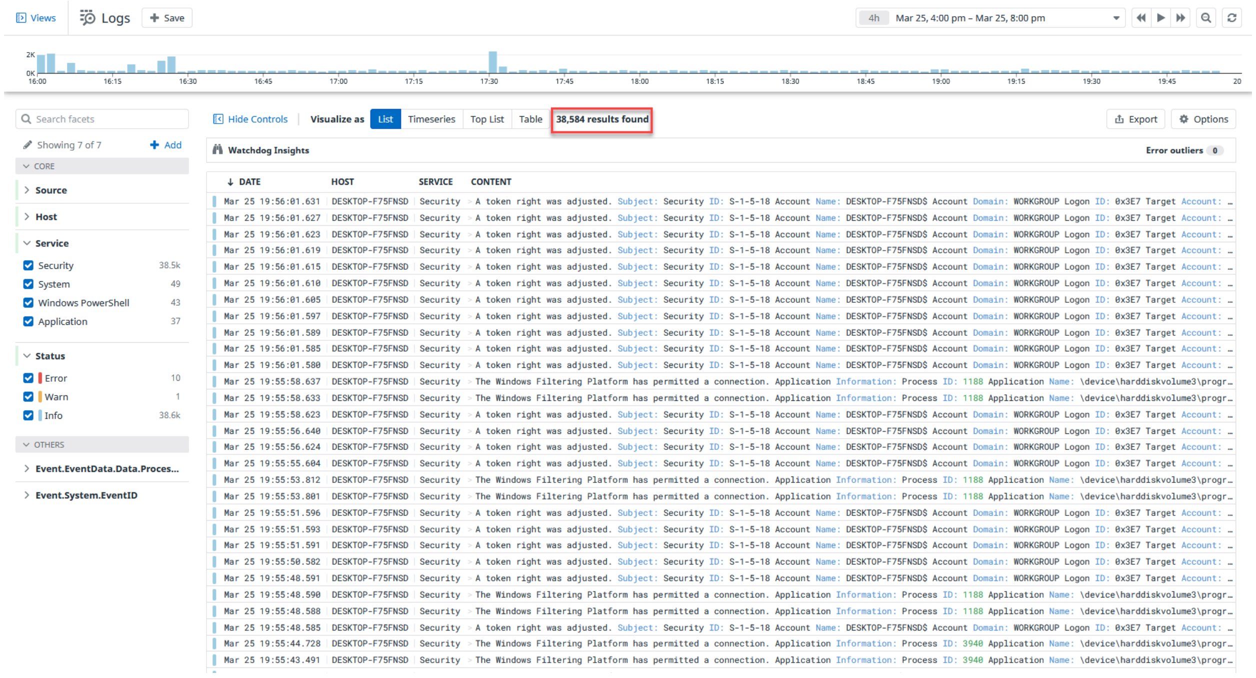
Data Presentation

Log File Size Reduction for Manual Inspection Simplification

As explained in chapter III, collecting logs, and forwarding them to a SIEM solution is fairly 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. Screenshot 4.1 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



38,584 results found

Showing 7 of 7 | Add

CORE
Source
Host
Service

✓ Security	38.5k
✓ System	49
✓ Windows PowerShell	43
✓ Application	37

Status

✓ Error	10
✓ Warn	1
✓ Info	38.6k

OTHERS
Event.EventData.Data.Proces...
Event.System.EventID

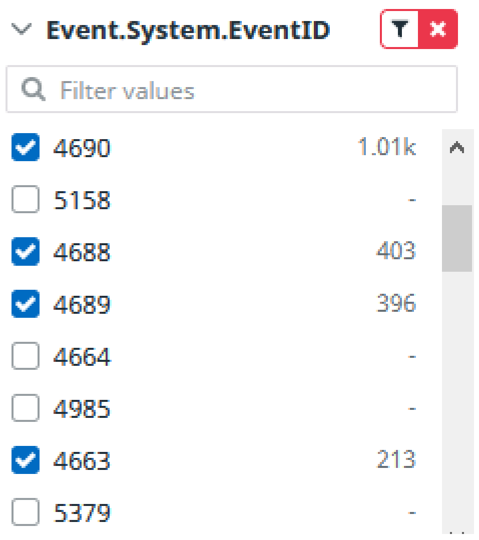
Watchdog Insights | Error outliers 0

```
DATE                HOST             SERVICE   CONTENT
Mar 25 19:56:01.631 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.627 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.623 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.619 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.615 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.610 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.605 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.597 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.589 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.585 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:56:01.580 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:58.637 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 1188 Application Name: \device\harddiskvolume3\progr…
Mar 25 19:55:58.633 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 1188 Application Name: \device\harddiskvolume3\progr…
Mar 25 19:55:58.623 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:56.640 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:56.624 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:55.604 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:53.812 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 1188 Application Name: \device\harddiskvolume3\progr…
Mar 25 19:55:53.801 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 1188 Application Name: \device\harddiskvolume3\progr…
Mar 25 19:55:51.596 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:51.593 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:51.591 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:50.582 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:48.591 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:48.590 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 1188 Application Name: \device\harddiskvolume3\progr…
Mar 25 19:55:48.588 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 1188 Application Name: \device\harddiskvolume3\progr…
Mar 25 19:55:48.585 DESKTOP-F75FNSD  Security  A token right was adjusted. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Target Account: …
Mar 25 19:55:44.728 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 3940 Application Name: \device\harddiskvolume3\progr…
Mar 25 19:55:43.491 DESKTOP-F75FNSD  Security  The Windows Filtering Platform has permitted a connection. Application Information: Process ID: 3940 Application Name: \device\harddiskvolume3\progr…
```

To narrow the scope of logs, filters were implemented on the DataDog SIEM to only include the Windows Security Event ID's that are 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 screenshot four. Furthermore, once the filter was applied, the number of events listed was reduced to 3,238. 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 number of filtered logs during a four-hour period containing an instance of ShadowMove



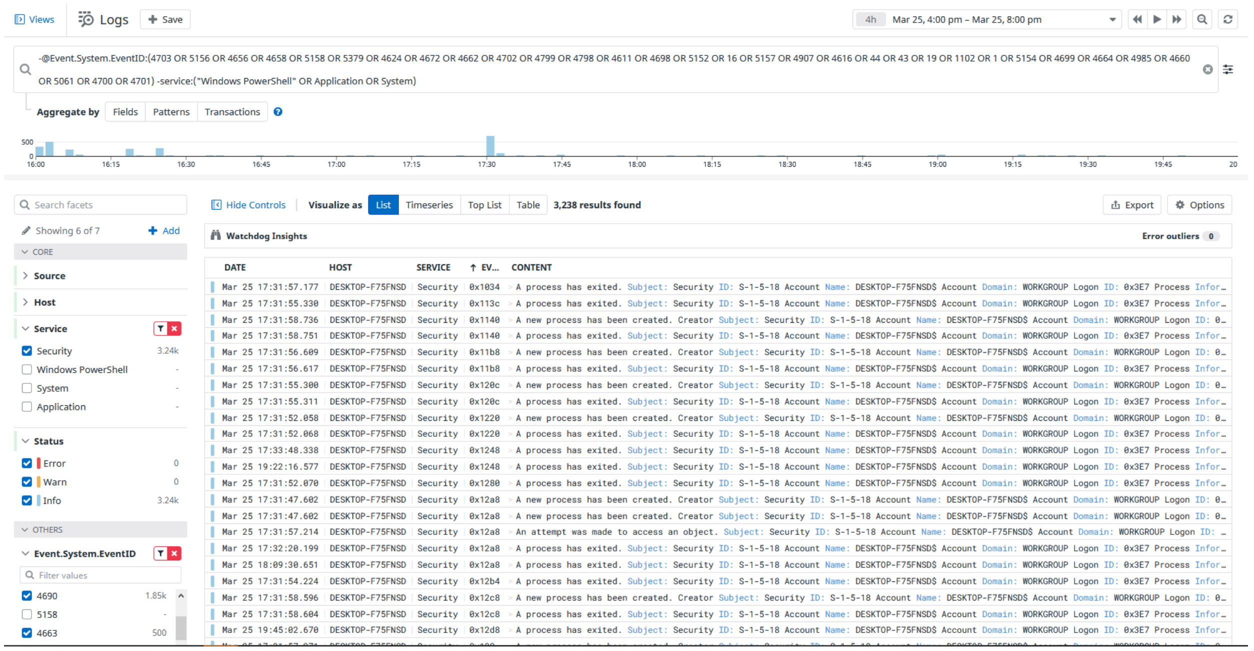
Event.System.EventID

Filter values

✓	4690	1.01k
☐	5158	-
✓	4688	403
✓	4689	396
☐	4664	-
☐	4985	-
✓	4663	213
☐	5379	-

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



Views | Logs | Save | 4h Mar 25, 4:00 pm – Mar 25, 8:00 pm

```
-@Event.System.EventID:(4703 OR 5156 OR 4656 OR 4658 OR 5158 OR 5379 OR 4624 OR 4672 OR 4662 OR 4702 OR 4799 OR 4798 OR 4611 OR 4698 OR 5152 OR 16 OR 5157 OR 4907 OR 4616 OR 44 OR 43 OR 19 OR 1102 OR 1 OR 5154 OR 4699 OR 4664 OR 4985 OR 4660
OR 5061 OR 4700 OR 4701) -service:("Windows PowerShell" OR Application OR System)
```

Aggregate by | Fields | Patterns | Transactions

Hide Controls | Visualize as | List | Timeseries | Top List | Table | 3,238 results found | Export | Options

Watchdog Insights | Error outliers 0

Search facets	
Showing 6 of 7	Add
CORE	
Source	
Host	
Service	
☑ Security	3.24k
☐ Windows PowerShell	-
☐ System	-
☐ Application	-
Status	
☑ Error	0
☑ Warn	0
☑ Info	3.24k
OTHERS	
Event.System.EventID	
Filter values	
☑ 4690	1.85k
☐ 5158	-
☑ 4663	500

```
DATE                HOST             SERVICE   ↑ EV...  CONTENT
Mar 25 17:31:57.177 DESKTOP-F75FNSD  Security  0x1034   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:55.330 DESKTOP-F75FNSD  Security  0x113c   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:58.736 DESKTOP-F75FNSD  Security  0x1140   A new process has been created. Creator Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0…
Mar 25 17:31:58.751 DESKTOP-F75FNSD  Security  0x1140   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:56.609 DESKTOP-F75FNSD  Security  0x11b8   A new process has been created. Creator Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0…
Mar 25 17:31:56.617 DESKTOP-F75FNSD  Security  0x11b8   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:55.300 DESKTOP-F75FNSD  Security  0x120c   A new process has been created. Creator Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0…
Mar 25 17:31:55.311 DESKTOP-F75FNSD  Security  0x120c   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:52.058 DESKTOP-F75FNSD  Security  0x1220   A new process has been created. Creator Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0…
Mar 25 17:31:52.068 DESKTOP-F75FNSD  Security  0x1220   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:33:48.338 DESKTOP-F75FNSD  Security  0x1248   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 19:22:16.577 DESKTOP-F75FNSD  Security  0x1248   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:52.070 DESKTOP-F75FNSD  Security  0x1280   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:47.602 DESKTOP-F75FNSD  Security  0x12a8   A new process has been created. Creator Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0…
Mar 25 17:31:47.602 DESKTOP-F75FNSD  Security  0x12a8   A new process has been created. Creator Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0…
Mar 25 17:31:57.214 DESKTOP-F75FNSD  Security  0x12a8   An attempt was made to access an object. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: …
Mar 25 17:32:20.199 DESKTOP-F75FNSD  Security  0x12a8   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 18:09:30.651 DESKTOP-F75FNSD  Security  0x12a8   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:54.224 DESKTOP-F75FNSD  Security  0x12b4   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 17:31:58.596 DESKTOP-F75FNSD  Security  0x12c8   A new process has been created. Creator Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0…
Mar 25 17:31:58.604 DESKTOP-F75FNSD  Security  0x12c8   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
Mar 25 19:45:02.670 DESKTOP-F75FNSD  Security  0x12d8   A process has exited. Subject: Security ID: S-1-5-18 Account Name: DESKTOP-F75FNSD$ Account Domain: WORKGROUP Logon ID: 0x3E7 Process Infor…
```

API Monitor and Static Analysis of API Calls.

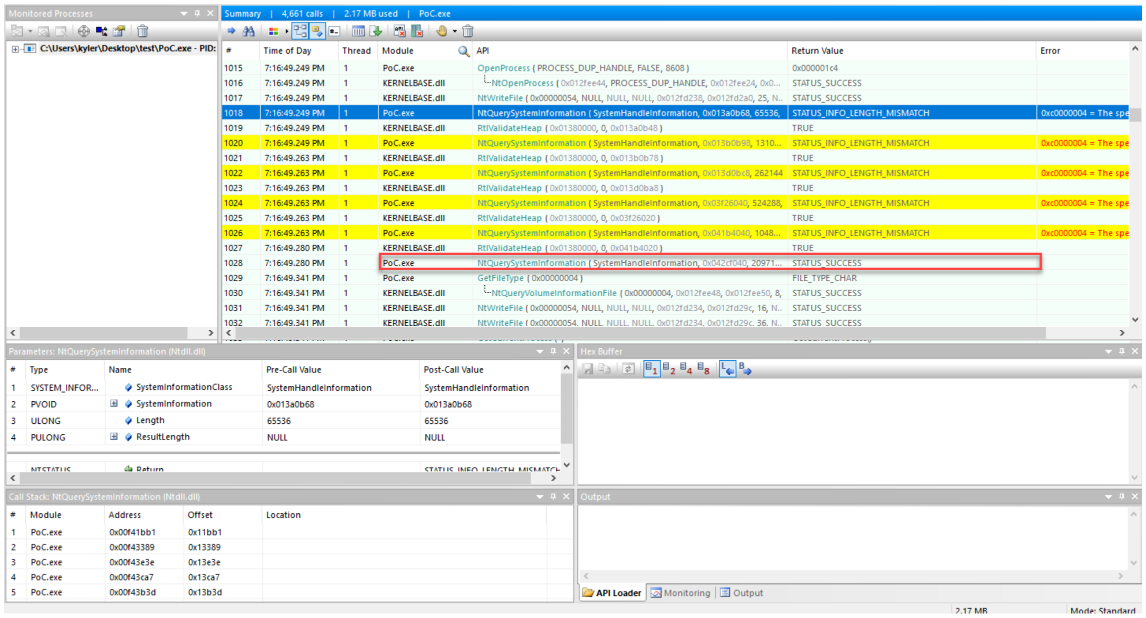
Manual inspection of the API Monitor output shows the ingenuity behind the ShadowMove attack. Each stage of the attack as outlined in Section III is caught during the execution of ShadowMove by API Monitor. This section contains screenshots of the API calls made during runtime and shows that successful socket duplication occurs and demonstrates that the packaged libraries are one of ShadowMove's greatest strengths and its biggest weakness. Something to note in screenshots four through seven 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 that it can inject into. On line 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


Monitored Processes
C:\Users\kyler\Desktop\test\PoC.exe - PID:
Summary | 4,661 calls | 2.17 MB used | PoC.exe
Hex Buffer
Output
API Loader
Monitoring
Output
2.17 MB
Mode: Standard

#	Time of Day	Thread	Module	API	Return Value	Error
1015	7:16:49.249 PM	1	PoC.exe	OpenProcess (PROCESS_DUP_HANDLE, FALSE, 8608)	0x000001c4	
1016	7:16:49.249 PM	1	KERNELBASE.dll	└NtOpenProcess (0x012fee44, PROCESS_DUP_HANDLE, 0x012fee24, 0x0...	STATUS_SUCCESS	
1017	7:16:49.249 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd238, 0x012fd2a0, 25, N..	STATUS_SUCCESS	
1018	7:16:49.249 PM	1	PoC.exe	NtQuerySystemInformation (SystemHandleInformation, 0x013a0b68, 65536,	STATUS_INFO_LENGTH_MISMATCH	0xc0000004 = The spe
1019	7:16:49.249 PM	1	KERNELBASE.dll	RtlValidateHeap (0x01380000, 0, 0x013a0b48)	TRUE	
1020	7:16:49.249 PM	1	PoC.exe	NtQuerySystemInformation (SystemHandleInformation, 0x013b0b98, 1310...	STATUS_INFO_LENGTH_MISMATCH	0xc0000004 = The spe
1021	7:16:49.263 PM	1	KERNELBASE.dll	RtlValidateHeap (0x01380000, 0, 0x013b0b78)	TRUE	
1022	7:16:49.263 PM	1	PoC.exe	NtQuerySystemInformation (SystemHandleInformation, 0x013d0bc8, 262144	STATUS_INFO_LENGTH_MISMATCH	0xc0000004 = The spe
1023	7:16:49.263 PM	1	KERNELBASE.dll	RtlValidateHeap (0x01380000, 0, 0x013d0ba8)	TRUE	
1024	7:16:49.263 PM	1	PoC.exe	NtQuerySystemInformation (SystemHandleInformation, 0x03f26040, 524288,	STATUS_INFO_LENGTH_MISMATCH	0xc0000004 = The spe
1025	7:16:49.263 PM	1	KERNELBASE.dll	RtlValidateHeap (0x01380000, 0, 0x03f26020)	TRUE	
1026	7:16:49.263 PM	1	PoC.exe	NtQuerySystemInformation (SystemHandleInformation, 0x041b4040, 1048...	STATUS_INFO_LENGTH_MISMATCH	0xc0000004 = The spe
1027	7:16:49.280 PM	1	KERNELBASE.dll	RtlValidateHeap (0x01380000, 0, 0x041b4020)	TRUE	
1028	7:16:49.280 PM	1	PoC.exe	NtQuerySystemInformation (SystemHandleInformation, 0x042cf040, 20971...	STATUS_SUCCESS	
1029	7:16:49.341 PM	1	PoC.exe	GetFileType (0x00000004)	FILE_TYPE_CHAR	
1030	7:16:49.341 PM	1	KERNELBASE.dll	└NtQueryVolumeInformationFile (0x00000004, 0x012fee48, 0x012fee50, 8,	STATUS_SUCCESS	
1031	7:16:49.341 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd234, 0x012fd29c, 16, N..	STATUS_SUCCESS	
1032	7:16:49.341 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd234, 0x012fd29c, 36, N..	STATUS_SUCCESS	

Parameters: NtQuerySystemInformation (Ntdll.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	SYSTEM_INFOR...	SystemInformationClass	SystemHandleInformation	SystemHandleInformation
2	PVOID	⊞ SystemInformation	0x013a0b68	0x013a0b68
3	ULONG	Length	65536	65536
4	PULONG	⊞ ResultLength	NULL	NULL
	NTSTATUS	Return		STATUS_INFO_LENGTH_MISMATCH

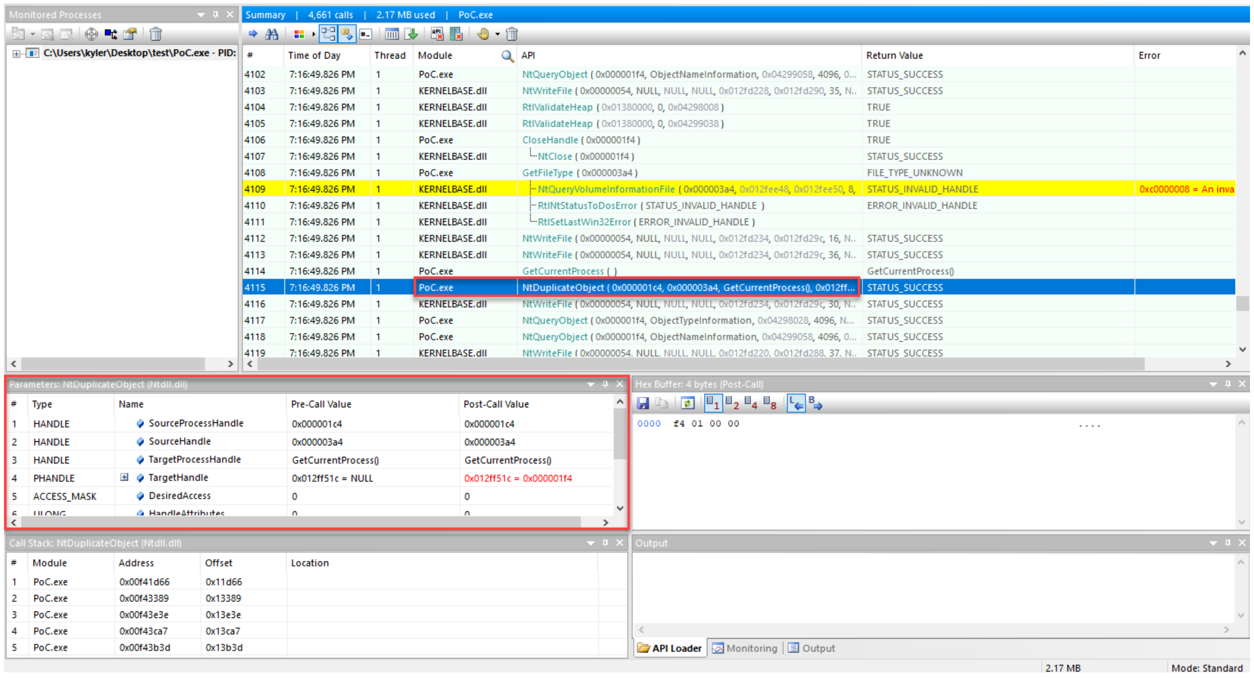
Call Stack: NtQuerySystemInformation (Ntdll.dll)

#	Module	Address	Offset	Location
1	PoC.exe	0x00f41bb1	0x11bb1	
2	PoC.exe	0x00f43389	0x13389	
3	PoC.exe	0x00f43e3e	0x13e3e	
4	PoC.exe	0x00f43ca7	0x13ca7	
5	PoC.exe	0x00f43b3d	0x13b3d	

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



Monitored Processes

C:\Users\kyler\Desktop\test\PoC.exe - PID:

Summary | 4,661 calls | 2.17 MB used | PoC.exe

#	Time of Day	Thread	Module	API	Return Value	Error
4102	7:16:49.826 PM	1	PoC.exe	NtQueryObject (0x000001f4, ObjectNameInformation, 0x04299058, 4096, 0...	STATUS_SUCCESS	
4103	7:16:49.826 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd228, 0x012fd290, 35, N..	STATUS_SUCCESS	
4104	7:16:49.826 PM	1	KERNELBASE.dll	RtlValidateHeap (0x01380000, 0, 0x04298008)	TRUE	
4105	7:16:49.826 PM	1	KERNELBASE.dll	RtlValidateHeap (0x01380000, 0, 0x04299038)	TRUE	
4106	7:16:49.826 PM	1	PoC.exe	CloseHandle (0x000001f4)	TRUE	
4107	7:16:49.826 PM	1	KERNELBASE.dll	NtClose (0x000001f4)	STATUS_SUCCESS	
4108	7:16:49.826 PM	1	PoC.exe	GetFileType (0x000003a4)	FILE_TYPE_UNKNOWN	
4109	7:16:49.826 PM	1	KERNELBASE.dll	NtQueryVolumeInformationFile (0x000003a4, 0x012fee48, 0x012fee50, 8,	STATUS_INVALID_HANDLE	0xc0000008 = An inva
4110	7:16:49.826 PM	1	KERNELBASE.dll	RtlNtStatusToDosError (STATUS_INVALID_HANDLE)	ERROR_INVALID_HANDLE	
4111	7:16:49.826 PM	1	KERNELBASE.dll	RtlSetLastWin32Error (ERROR_INVALID_HANDLE)		
4112	7:16:49.826 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd234, 0x012fd29c, 16, N..	STATUS_SUCCESS	
4113	7:16:49.826 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd234, 0x012fd29c, 36, N..	STATUS_SUCCESS	
4114	7:16:49.826 PM	1	PoC.exe	GetCurrentProcess ( )	GetCurrentProcess()	
4115	7:16:49.826 PM	1	PoC.exe	NtDuplicateObject (0x000001c4, 0x000003a4, GetCurrentProcess(), 0x012ff...	STATUS_SUCCESS	
4116	7:16:49.826 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd234, 0x012fd29c, 30, N..	STATUS_SUCCESS	
4117	7:16:49.826 PM	1	PoC.exe	NtQueryObject (0x000001f4, ObjectTypeInformation, 0x04298028, 4096, N...	STATUS_SUCCESS	
4118	7:16:49.826 PM	1	PoC.exe	NtQueryObject (0x000001f4, ObjectNameInformation, 0x04299058, 4096, 0...	STATUS_SUCCESS	
4119	7:16:49.826 PM	1	KERNELBASE.dll	NtWriteFile (0x00000054, NULL, NULL, NULL, 0x012fd220, 0x012fd288, 37, N..	STATUS_SUCCESS	

Parameters: NtDuplicateObject (Ntdll.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	HANDLE	SourceProcessHandle	0x000001c4	0x000001c4
2	HANDLE	SourceHandle	0x000003a4	0x000003a4
3	HANDLE	TargetProcessHandle	GetCurrentProcess()	GetCurrentProcess()
4	PHANDLE	TargetHandle	0x012ff51c = NULL	0x012ff51c = 0x000001f4
5	ACCESS_MASK	DesiredAccess	0	0
6	ULONG	HandleAttributes	0	0

Hex Buffer: 4 bytes (Post-Call)

```
0000  f4 01 00 00                                   ....
```

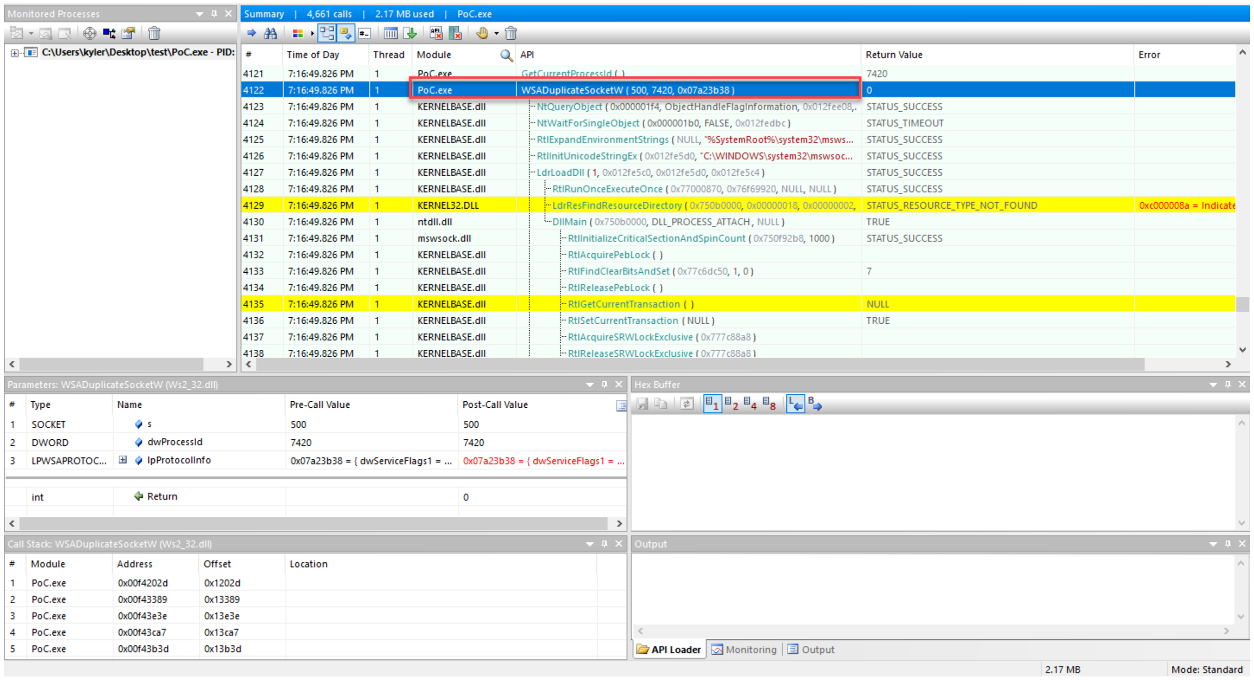
Call Stack: NtDuplicateObject (Ntdll.dll)

#	Module	Address	Offset	Location
1	PoC.exe	0x00f41d66	0x11d66	
2	PoC.exe	0x00f43389	0x13389	
3	PoC.exe	0x00f43e3e	0x13e3e	
4	PoC.exe	0x00f43ca7	0x13ca7	
5	PoC.exe	0x00f43b3d	0x13b3d	

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.

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



Monitored Processes

C:\Users\kyler\Desktop\test\PoC.exe - PID:

Summary | 4,661 calls | 2.17 MB used | PoC.exe

#	Time of Day	Thread	Module	API	Return Value	Error
4121	7:16:49.826 PM	1	PoC.exe	GetCurrentProcessId ( )	7420	
4122	7:16:49.826 PM	1	PoC.exe	WSADuplicateSocketW (500, 7420, 0x07a23b38)	0	
4123	7:16:49.826 PM	1	KERNELBASE.dll	NtQueryObject (0x000001f4, ObjectHandleFlagInformation, 0x012fee08,.	STATUS_SUCCESS	
4124	7:16:49.826 PM	1	KERNELBASE.dll	NtWaitForSingleObject (0x000001b0, FALSE, 0x012fedbc)	STATUS_TIMEOUT	
4125	7:16:49.826 PM	1	KERNELBASE.dll	RtlExpandEnvironmentStrings (NULL, "%SystemRoot%\system32\msws...	STATUS_SUCCESS	
4126	7:16:49.826 PM	1	KERNELBASE.dll	RtlInitUnicodeStringEx (0x012fe5d0, "C:\WINDOWS\system32\mswsoc...	STATUS_SUCCESS	
4127	7:16:49.826 PM	1	KERNELBASE.dll	LdrLoadDll (1, 0x012fe5c0, 0x012fe5d0, 0x012fe5c4)	STATUS_SUCCESS	
4128	7:16:49.826 PM	1	KERNELBASE.dll	RtlRunOnceExecuteOnce (0x77000870, 0x76f69920, NULL, NULL)	STATUS_SUCCESS	
4129	7:16:49.826 PM	1	KERNEL32.DLL	LdrResFindResourceDirectory (0x750b0000, 0x00000018, 0x00000002,	STATUS_RESOURCE_TYPE_NOT_FOUND	0xc000008a = Indicate
4130	7:16:49.826 PM	1	ntdll.dll	DllMain (0x750b0000, DLL_PROCESS_ATTACH, NULL)	TRUE	
4131	7:16:49.826 PM	1	mswsock.dll	RtlInitializeCriticalSectionAndSpinCount (0x750f92b8, 1000)	STATUS_SUCCESS	
4132	7:16:49.826 PM	1	KERNELBASE.dll	RtlAcquirePebLock ( )		
4133	7:16:49.826 PM	1	KERNELBASE.dll	RtlFindClearBitsAndSet (0x77c6dc50, 1, 0)	7	
4134	7:16:49.826 PM	1	KERNELBASE.dll	RtlReleasePebLock ( )		
4135	7:16:49.826 PM	1	KERNELBASE.dll	RtlGetCurrentTransaction ( )	NULL	
4136	7:16:49.826 PM	1	KERNELBASE.dll	RtlSetCurrentTransaction (NULL)	TRUE	
4137	7:16:49.826 PM	1	KERNELBASE.dll	RtlAcquireSRWLockExclusive (0x777c88a8)		
4138	7:16:49.826 PM	1	KERNELBASE.dll	RtlReleaseSRWLockExclusive (0x777c88a8)		

Parameters: WSADuplicateSocketW (Ws2_32.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	SOCKET	s	500	500
2	DWORD	dwProcessId	7420	7420
3	LPWSAPROTOC...	lpProtocolInfo	0x07a23b38 = { dwServiceFlags1 = ...	0x07a23b38 = { dwServiceFlags1 = ...
	int	Return		0

Hex Buffer

Call Stack: WSADuplicateSocketW (Ws2_32.dll)

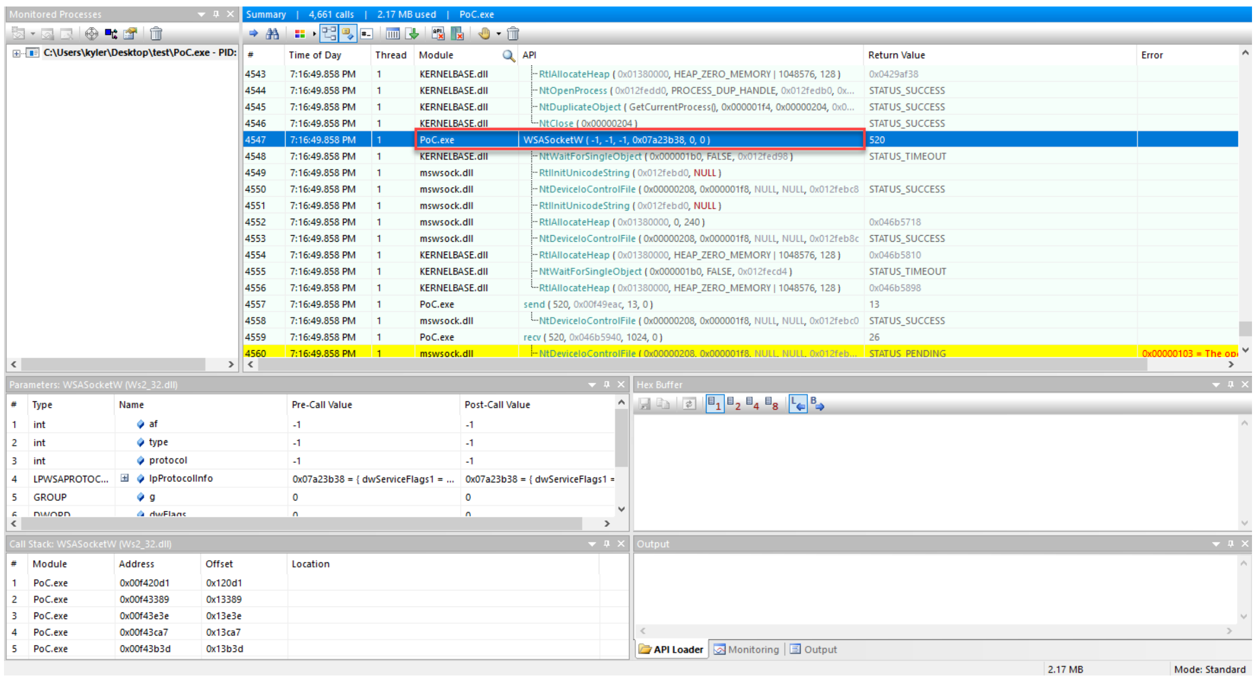
#	Module	Address	Offset	Location
1	PoC.exe	0x00f4202d	0x1202d	
2	PoC.exe	0x00f43389	0x13389	
3	PoC.exe	0x00f43e3e	0x13e3e	
4	PoC.exe	0x00f43ca7	0x13ca7	
5	PoC.exe	0x00f43b3d	0x13b3d	

Output

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 provided by WSADuplicateSocketW to connect to the duplicated socket. The socket connection takes place on line 4547 and the data send 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



Monitored Processes

C:\Users\kyler\Desktop\test\PoC.exe - PID:

Summary | 4,661 calls | 2.17 MB used | PoC.exe

#	Time of Day	Thread	Module	API	Return Value	Error
4543	7:16:49.858 PM	1	KERNELBASE.dll	RtlAllocateHeap (0x01380000, HEAP_ZERO_MEMORY | 1048576, 128)	0x0429af38	
4544	7:16:49.858 PM	1	KERNELBASE.dll	NtOpenProcess (0x012fedd0, PROCESS_DUP_HANDLE, 0x012fedb0, 0x...	STATUS_SUCCESS	
4545	7:16:49.858 PM	1	KERNELBASE.dll	NtDuplicateObject (GetCurrentProcess(), 0x000001f4, 0x00000204, 0x0...	STATUS_SUCCESS	
4546	7:16:49.858 PM	1	KERNELBASE.dll	NtClose (0x00000204)	STATUS_SUCCESS	
4547	7:16:49.858 PM	1	PoC.exe	WSASocketW (-1, -1, -1, 0x07a23b38, 0, 0)	520	
4548	7:16:49.858 PM	1	KERNELBASE.dll	NtWaitForSingleObject (0x000001b0, FALSE, 0x012fed98)	STATUS_TIMEOUT	
4549	7:16:49.858 PM	1	mswsock.dll	RtlInitUnicodeString (0x012febd0, NULL)		
4550	7:16:49.858 PM	1	mswsock.dll	NtDeviceIoControlFile (0x00000208, 0x000001f8, NULL, NULL, 0x012febc8	STATUS_SUCCESS	
4551	7:16:49.858 PM	1	mswsock.dll	RtlInitUnicodeString (0x012febd0, NULL)		
4552	7:16:49.858 PM	1	mswsock.dll	RtlAllocateHeap (0x01380000, 0, 240)	0x046b5718	
4553	7:16:49.858 PM	1	mswsock.dll	NtDeviceIoControlFile (0x00000208, 0x000001f8, NULL, NULL, 0x012feb8c	STATUS_SUCCESS	
4554	7:16:49.858 PM	1	KERNELBASE.dll	RtlAllocateHeap (0x01380000, HEAP_ZERO_MEMORY | 1048576, 128)	0x046b5810	
4555	7:16:49.858 PM	1	KERNELBASE.dll	NtWaitForSingleObject (0x000001b0, FALSE, 0x012fecd4)	STATUS_TIMEOUT	
4556	7:16:49.858 PM	1	KERNELBASE.dll	RtlAllocateHeap (0x01380000, HEAP_ZERO_MEMORY | 1048576, 128)	0x046b5898	
4557	7:16:49.858 PM	1	PoC.exe	send (520, 0x00f49eac, 13, 0)	13	
4558	7:16:49.858 PM	1	mswsock.dll	NtDeviceIoControlFile (0x00000208, 0x000001f8, NULL, NULL, 0x012febc0	STATUS_SUCCESS	
4559	7:16:49.858 PM	1	PoC.exe	recv (520, 0x046b5940, 1024, 0)	26	
4560	7:16:49.858 PM	1	mswsock.dll	NtDeviceIoControlFile (0x00000208, 0x000001f8, NULL, NULL, 0x012feb...	STATUS_PENDING	0x00000103 = The op

Parameters: WSASocketW (Ws2_32.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	int	af	-1	-1
2	int	type	-1	-1
3	int	protocol	-1	-1
4	LPWSAPROTOC...	lpProtocolInfo	0x07a23b38 = { dwServiceFlags1 = ...	0x07a23b38 = { dwServiceFlags1 =
5	GROUP	g	0	0
6	DWORD	dwFlags	0	0

Hex Buffer

Call Stack: WSASocketW (Ws2_32.dll)

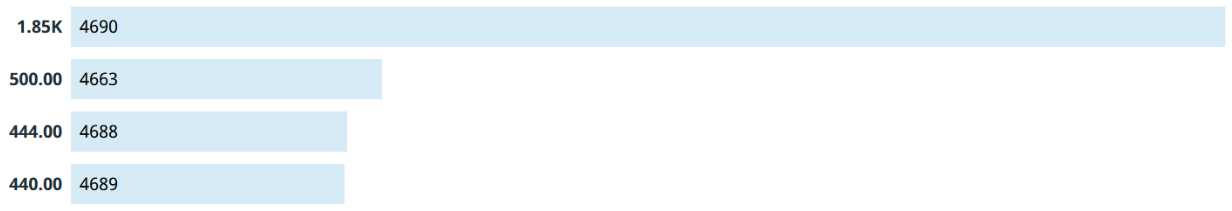
#	Module	Address	Offset	Location
1	PoC.exe	0x00f420d1	0x120d1	
2	PoC.exe	0x00f43389	0x13389	
3	PoC.exe	0x00f43e3e	0x13e3e	
4	PoC.exe	0x00f43ca7	0x13ca7	
5	PoC.exe	0x00f43b3d	0x13b3d	

Output

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. If a suspicious program name or location is found to be duplicating handles, then 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 that are relevant to the investigation.

Figure 9

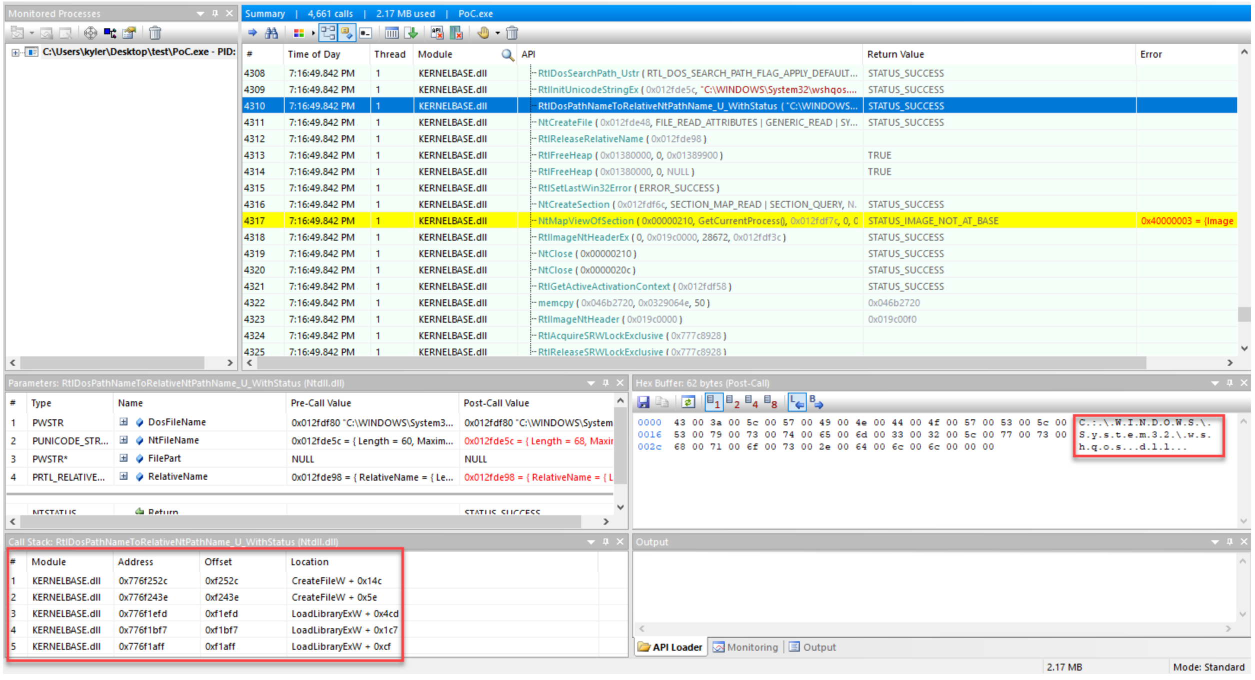
A large number of handle duplication events takes place every hour on a Windows system



The final screenshot 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 for 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



Monitored Processes

C:\Users\kyler\Desktop\test\PoC.exe - PID:

Summary | 4,661 calls | 2.17 MB used | PoC.exe

#	Time of Day	Thread	Module	API	Return Value	Error
4308	7:16:49.842 PM	1	KERNELBASE.dll	RtlDosSearchPath_Ustr (RTL_DOS_SEARCH_PATH_FLAG_APPLY_DEFAULT...	STATUS_SUCCESS	
4309	7:16:49.842 PM	1	KERNELBASE.dll	RtlInitUnicodeStringEx (0x012fde5c, "C:\WINDOWS\System32\wshqos....	STATUS_SUCCESS	
4310	7:16:49.842 PM	1	KERNELBASE.dll	RtlDosPathNameToRelativeNtPathName_U_WithStatus ("C:\WINDOWS...	STATUS_SUCCESS	
4311	7:16:49.842 PM	1	KERNELBASE.dll	NtCreateFile (0x012fde48, FILE_READ_ATTRIBUTES | GENERIC_READ | SY...	STATUS_SUCCESS	
4312	7:16:49.842 PM	1	KERNELBASE.dll	RtlReleaseRelativeName (0x012fde98)		
4313	7:16:49.842 PM	1	KERNELBASE.dll	RtlFreeHeap (0x01380000, 0, 0x01389900)	TRUE	
4314	7:16:49.842 PM	1	KERNELBASE.dll	RtlFreeHeap (0x01380000, 0, NULL)	TRUE	
4315	7:16:49.842 PM	1	KERNELBASE.dll	RtlSetLastWin32Error (ERROR_SUCCESS)		
4316	7:16:49.842 PM	1	KERNELBASE.dll	NtCreateSection (0x012fdf6c, SECTION_MAP_READ | SECTION_QUERY, N.	STATUS_SUCCESS	
4317	7:16:49.842 PM	1	KERNELBASE.dll	NtMapViewOfSection (0x00000210, GetCurrentProcess(), 0x012fdf7c, 0, 0	STATUS_IMAGE_NOT_AT_BASE	0x40000003 = {Image
4318	7:16:49.842 PM	1	KERNELBASE.dll	RtlImageNtHeaderEx (0, 0x019c0000, 28672, 0x012fdf3c)	STATUS_SUCCESS	
4319	7:16:49.842 PM	1	KERNELBASE.dll	NtClose (0x00000210)	STATUS_SUCCESS	
4320	7:16:49.842 PM	1	KERNELBASE.dll	NtClose (0x0000020c)	STATUS_SUCCESS	
4321	7:16:49.842 PM	1	KERNELBASE.dll	RtlGetActiveActivationContext (0x012fdf58)	STATUS_SUCCESS	
4322	7:16:49.842 PM	1	KERNELBASE.dll	memcpy (0x046b2720, 0x0329064e, 50)	0x046b2720	
4323	7:16:49.842 PM	1	KERNELBASE.dll	RtlImageNtHeader (0x019c0000)	0x019c00f0	
4324	7:16:49.842 PM	1	KERNELBASE.dll	RtlAcquireSRWLockExclusive (0x777c8928)		
4325	7:16:49.842 PM	1	KERNELBASE.dll	RtlReleaseSRWLockExclusive (0x777c8928)		

Parameters: RtlDosPathNameToRelativeNtPathName_U_WithStatus (Ntdll.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	PWSTR	DosFileName	0x012fdf80 "C:\WINDOWS\System3...	0x012fdf80 "C:\WINDOWS\System
2	PUNICODE_STR...	NtFileName	0x012fde5c = { Length = 60, Maxim...	0x012fde5c = { Length = 68, Maxir
3	PWSTR*	FilePart	NULL	NULL
4	PRTL_RELATIVE...	RelativeName	0x012fde98 = { RelativeName = { Le...	0x012fde98 = { RelativeName = { L
	NTSTATUS	Return		STATUS_SUCCESS

Hex Buffer: 62 bytes (Post-Call)

```
0000  43 00 3a 00 5c 00 57 00 49 00 4e 00 44 00 4f 00 57 00 53 00 5c 00  C.:.\.W.I.N.D.O.W.S.\.
0016  53 00 79 00 73 00 74 00 65 00 6d 00 33 00 32 00 5c 00 77 00 73 00  S.y.s.t.e.m.3.2.\.w.s.
002c  68 00 71 00 6f 00 73 00 2e 00 64 00 6c 00 6c 00 00 00              h.q.o.s...d.l.l...
```

Call Stack: RtlDosPathNameToRelativeNtPathName_U_WithStatus (Ntdll.dll)

#	Module	Address	Offset	Location
1	KERNELBASE.dll	0x776f252c	0xf252c	CreateFileW + 0x14c
2	KERNELBASE.dll	0x776f243e	0xf243e	CreateFileW + 0x5e
3	KERNELBASE.dll	0x776f1efd	0xf1efd	LoadLibraryExW + 0x4cd
4	KERNELBASE.dll	0x776f1bf7	0xf1bf7	LoadLibraryExW + 0x1c7
5	KERNELBASE.dll	0x776f1aff	0xf1aff	LoadLibraryExW + 0xcf

Output

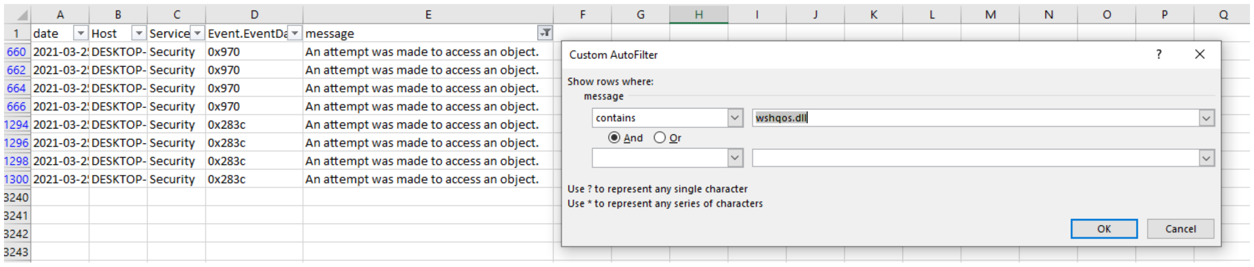
Data Analysis

Because ShadowMove touches so little of the core operating system, handle 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, I 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 I propose will allow the analyst to narrow down the list of suspect processes to a level where manual analysis is possible.

The method for analyzing the data 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. This 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



	A	B	C	D	E
1	date	Host	Service	Event.EventDa	message
660	2021-03-2!	DESKTOP-	Security	0x970	An attempt was made to access an object.
662	2021-03-2!	DESKTOP-	Security	0x970	An attempt was made to access an object.
664	2021-03-2!	DESKTOP-	Security	0x970	An attempt was made to access an object.
666	2021-03-2!	DESKTOP-	Security	0x970	An attempt was made to access an object.
1294	2021-03-2!	DESKTOP-	Security	0x283c	An attempt was made to access an object.
1296	2021-03-2!	DESKTOP-	Security	0x283c	An attempt was made to access an object.
1298	2021-03-2!	DESKTOP-	Security	0x283c	An attempt was made to access an object.
1300	2021-03-2!	DESKTOP-	Security	0x283c	An attempt was made to access an object.
3240					
3241					
3242					
3243					

Custom AutoFilter

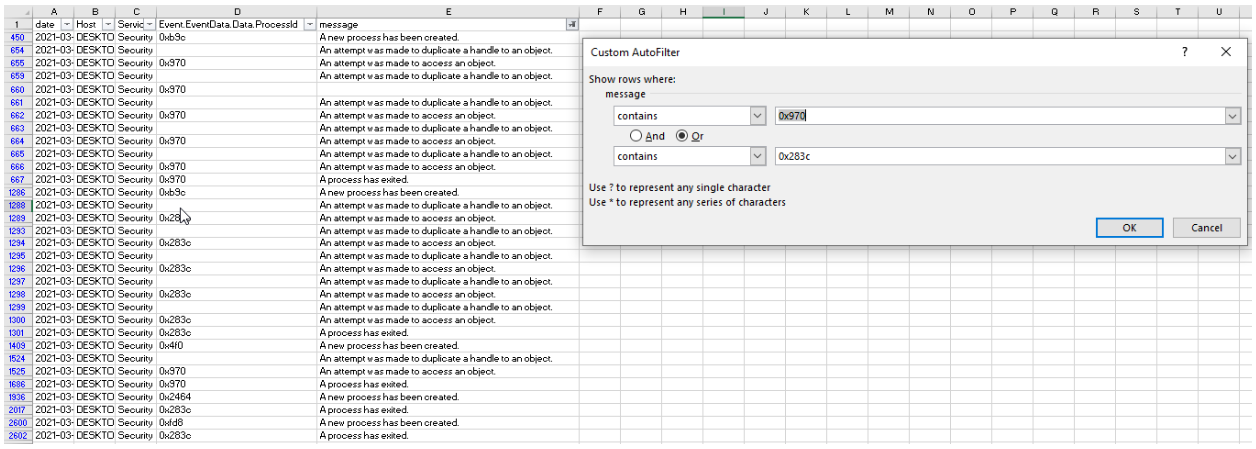
Show rows where:
message
contains — wshqos.dll
[✓ And] Or

Use ? to represent any single character
Use * to represent any series of characters

These log messages provide detailed information on the process name that 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 the execution of both processes. An analyst, at this point, would note that the process was created, attempted to duplicate objects and access objects that are being audited, and then exit. This follows the operating procedure of ShadowMove and upon closer inspection if the process name is unknown or it is running from a strange directory, it is likely that an unwanted program is executing 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



	date	Host	Servic	Event.EventData.Data.ProcessId	message
450	2021-03-	DESKTO	Security	0xb9c	A new process has been created.
654	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
655	2021-03-	DESKTO	Security	0x970	An attempt was made to access an object.
659	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
660	2021-03-	DESKTO	Security	0x970	
661	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
662	2021-03-	DESKTO	Security	0x970	An attempt was made to access an object.
663	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
664	2021-03-	DESKTO	Security	0x970	An attempt was made to access an object.
665	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
666	2021-03-	DESKTO	Security	0x970	An attempt was made to access an object.
667	2021-03-	DESKTO	Security	0x970	A process has exited.
1286	2021-03-	DESKTO	Security	0xb9c	A new process has been created.
1288	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
1289	2021-03-	DESKTO	Security	0x28	An attempt was made to access an object.
1293	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
1294	2021-03-	DESKTO	Security	0x283c	An attempt was made to access an object.
1295	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
1296	2021-03-	DESKTO	Security	0x283c	An attempt was made to access an object.
1297	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
1298	2021-03-	DESKTO	Security	0x283c	An attempt was made to access an object.
1299	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
1300	2021-03-	DESKTO	Security	0x283c	An attempt was made to access an object.
1301	2021-03-	DESKTO	Security	0x283c	A process has exited.
1409	2021-03-	DESKTO	Security	0x4f0	A new process has been created.
1524	2021-03-	DESKTO	Security		An attempt was made to duplicate a handle to an object.
1525	2021-03-	DESKTO	Security	0x970	An attempt was made to access an object.
1686	2021-03-	DESKTO	Security	0x970	A process has exited.
1936	2021-03-	DESKTO	Security	0x2464	A new process has been created.
2017	2021-03-	DESKTO	Security	0x283c	A process has exited.
2600	2021-03-	DESKTO	Security	0xfd8	A new process has been created.
2602	2021-03-	DESKTO	Security	0x283c	A process has exited.

Custom AutoFilter

Show rows where:
message
contains 0x970
( ) And (●) Or
contains 0x283c

Use ? to represent any single character
Use * to represent any series of characters

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

Summary

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. 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 it is possible that a ShadowMove has occurred. By filtering in this manner, an analyst would be able to find file names and process ids to further investigate, however, without statically or dynamically analyzing the code, it would be impossible to determine with complete certainty that ShadowMove occurred.

Chapter V: Results, Conclusion, and Recommendations

Introduction

In this paper, I introduced traditional methods for lateral movement in Windows systems as well as well-known defenses for protecting systems from malicious lateral movement. Likewise, I explain why lateral movement is such a persistent issue and postulate on the fact 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. I took the time to research new methods proposed for lateral movement detection including graph-based and machine learning-based models. Finally, I presented ShadowMove, how it functions, and a new method for detecting ShadowMove which has not been detectable to my knowledge.

Results

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 I propose will lead an analyst to an executable file for manual or dynamic code analysis.

I confirmed that ShadowMove is a legitimate threat and is excellent at evading detection as it hardly touches the host operating system. I was 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, I was able to develop 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.

Conclusion

The key to ensuring success in detecting a ShadowMove lies in auditing the correct files. 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 thought to be 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 Work

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 of the detection process. This is a ruleset that I 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 commit.

Another area I 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 my research lab was only a single virtual machine running extraordinarily little software, it is possible that this method may generate more false positives than I anticipate. I 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 that are 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 took place using specific function calls from dynamic link libraries.

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, I believe it will be important in the future to always monitor specific function calls from dynamic link libraries. 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 R. Holt and his team could lead to significantly more secure systems (Holt et al., 2019).

References

Abe, S. (2016). *Detecting Lateral Movement in APTs~Analysis Approach on Windows Event Logs~*. JPCERT/CC.

API Monitor - Spy and display Win32 API calls made by applications. (2013). Apimonitor.com. https://apimonitor.com/

Bai, T., Bian, H., Abou Daya, A., Salahuddin, M., Limam, N., & Boutaba, R. (2019). *A Machine Learning Approach for RDP-based Lateral Movement Detection*. 242–245.

Bohara, A., Noureddine, M., Fawaz, A., & Sanders, W. (2017). An Unsupervised Multi-Detector Approach for Identifying Malicious Lateral Movement. *2017 IEEE 36th Symposium on Reliable Distributed Systems*.

Chandrasekhar, A. M., & Raghuveer, K. (2013). *Intrusion Detection Technique by using K-means, Fuzzy Neural Network and SVM classifiers.*

Chen, H., & Jiang, L. (2019). Efficient GAN-based method for cyber-intrusion detection. *Cs.LG*.

CVE-2017-0143: The SMBv1 server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and R2 SP1; Windows 7 SP1; Windows 8.1; Windows. (n.d.). Www.cvedetails.com. Retrieved March 31, 2021, from https://www.cvedetails.com/cve/CVE-2017-0143

Datadog. (2019). *Modern monitoring & analytics* | *Datadog*. Modern Monitoring & Analytics. https://www.datadoghq.com/

Datadog. (2021a). *Agent*. Datadog Infrastructure and Application Monitoring. https://docs.datadoghq.com/agent

Datadog. (2021b). *Getting Started with Datadog*. Datadog Infrastructure and Application Monitoring. https://docs.datadoghq.com/integrations/active_directory/

Dulkin, A., Sade, Y., Benedict, O., Stanford, J., & Lazarovitz, L. (2017). *SYSTEMS AND METHODS FOR DETECTING AND REACTING TO MALICIOUS ACTIVITY IN COMPUTER NETWORKS* (United States Patent and Trademark Office. Patent).

https://patentimages.storage.googleapis.com/df/ee/0c/f6ea400c88751f/US20170257376A1.pdf

Fujimoto, M., Matsuda, W., & Mitsunaga, T. (2018). Detecting Abuse of Domain Administrator privilege using Windows Event Logs. *2018 IEEE Conference on Applications, Information and Network Security (AINS)*.

Gogoi, P., Bhattacharyya, D. K., Borah, B., & Kalita, J. K. (2013). MLH-IDS: A Multi-Level Hybrid Intrusion Detection Method. *The Computer Journal*, *57*(4), 602–623.https://doi.org/10.1093/comjnl/bxt044

Holt, R., Aubrey, S., DeVille, A., Haight, W., Gary, T., & Wang, Q. (2019). Deep Autoencoder Neural Networks for Detecting Lateral Movement in Computer Networks. *International Conference on Artificial Intelligence*.

Hudek, T., Varadharajan, K., Viviano, A., Graf, E., Cymoki, Komsorg, Prashantchahar, Resnik, S., Jacobs, M., Iudezgit, & MacMichael, D. (2020, August 17). *Download the Windows Driver Kit (WDK) - Windows drivers*. Docs.microsoft.com. https://docs.microsoft.com/en-us/windows-hardware/drivers/download-the-wdk

Kaiafas, G., Varisteas, G., Lagraa, S., State, R., Nguyen, C. D., Ries, T., & Ourdane, M. (2018). Detecting Malicious Authentication Events Trustfully. *IEEE*. IEEE.

Lateral Movement, Tactic TA0008 - Enterprise | MITRE ATT&CK®. (2018, October 17). Attack.mitre.org. https://attack.mitre.org/tactics/TA0008/

Liang, H., Rugerio, D., Chen, L., & Xu, S. (2020, September 22). *Dynamic link library (DLL) - Windows Client*. Docs.microsoft.com. https://docs.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library

Liu, H., & Lang, B. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Applied Sciences*, *9*(20),4396. https://doi.org/10.3390/app9204396

Liu, Q., Stokes, J., Mead, R., Burrell, T., Hellen, I., Lambert, J., Marochko, A., & Cui, W. (2018). Latte: Large-Scale Lateral Movement Detection. *Milcom 2018 Track 3 - Cyber Security and Trusted Computing,* IEEE.

Lynch, V. (2017, May 26). *Cost of 2013 Target Data Breach Nears $300 Million*. Hashed out by the SSL Store^{TM}. https://www.thesslstore.com/blog/2013-target-data-breach-settled/

Marshall, D., Coulter, D., Bazan, N., & Graff, E. (2018, June 28). *Driver Development Tools - Windows drivers*. Docs.microsoft.com. https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest

Matrix - Enterprise | *MITRE ATT&CK®*. (2020, October 27). Attack.mitre.org; MITRE. https://attack.mitre.org/matrices/enterprise/windows/

Niakanlahiji, A., Wei, J., Alam, R., Wang, Q., & Chu, B.-T. (2020). ShadowMove: A Stealthy Lateral Movement Strategy. *The 29th USENIX Security Symposium*.

Oh, J. (2016.). *A Forensic Analysis of APT Lateral Movement in Windows Environment*. AhnLab.

Pedemakar, P. (2020, February 2). *What is SMB?* | *How it Works* | *Features & authentication protocol of SMB*. EDUCBA. https://www.educba.com/what-is-smb/

Pei Liew, S., & Ikeda, S. (2019). Detecting Adversary using Windows Digital Artifacts. *2019 IEEE International Conference on Big Data (Big Data)*. 2019 IEEE International Conference on Big Data (Big Data).

Pektaş, A., & Basaranoglu, E. (2017). Practical Approach for Securing Windows Environment: Attack Vectors and Countermeasures. *SSRN Electronic Journal*, *9*(6). https://doi.org/10.2139/ssrn.3649907

Plotnik, I., Be’ery, T. A., Dolinsky, M., Plotnik, O., Messerman, G., & Krigsman, S. (2017). *System, method, and process for detecting advanced and targeted attacks with the recoupling of Kerberos authentication and authorization* (The United States Patent and Trademark Office Patent). https://patentimages.storage.googleapis.com/63/64/de/37cfff6ee6daf8/US9729538.pdf

Powell, B. A. (2019). The epidemiology of lateral movement: exposures and countermeasures with network contagion models. *Journal of Cyber Security Technology*, 1–39. https://doi.org/10.1080/23742917.2019.1627702

Schofield, M., Coulter, D., & Satran, M. (2018, May 31). *Remote Desktop Protocol - Win32 apps*. Microsoft.com. https://docs.microsoft.com/en-us/windows/win32/termserv/remote-desktop-protocol

Shelley, S. (2019, June 27). *Phishing Number One Cause of Data Breaches: Lessons from Verizon DBIR*. Info.phishlabs.com. https://info.phishlabs.com/blog/phishing-number-1-data-breaches-lessons-verizon

Siadati, H., & Memon, N. (2017). Detecting Structurally Anomalous Logins Within Enterprise Networks. *Insights from Log(In)S*, 1273–1284. CCS’17.

Sidati, H., Saket, B., & Memon, N. (2016). Detecting Malicious Logins in Enterprise Networks Using Visualization. *IEEE*. IEEE.

Simpson, D., Yoshioka, H., Avedon, M. H., Hall, J., Gorzelany, A. M., & Bischel, A. (2017a, April 19). *Audit object access (Windows 10) - Windows security*. Docs.microsoft.com. https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/basic-audit-object-access

Simpson, D., Onur, Hall, J., Gorzelany, A. M., & Schonning, N. (2017b, April 19). *Audit process tracking (Windows 10) - Windows security*. Docs.microsoft.com. https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/basic-audit-process-tracking

Tomonaga, S. (2016, January 26). *Windows Commands Abused by Attackers*. JPCERT/CC Eyes; JPCERT/CC. https://blogs.jpcert.or.jp/en/2016/01/windows-commands-abused-by-attackers.html

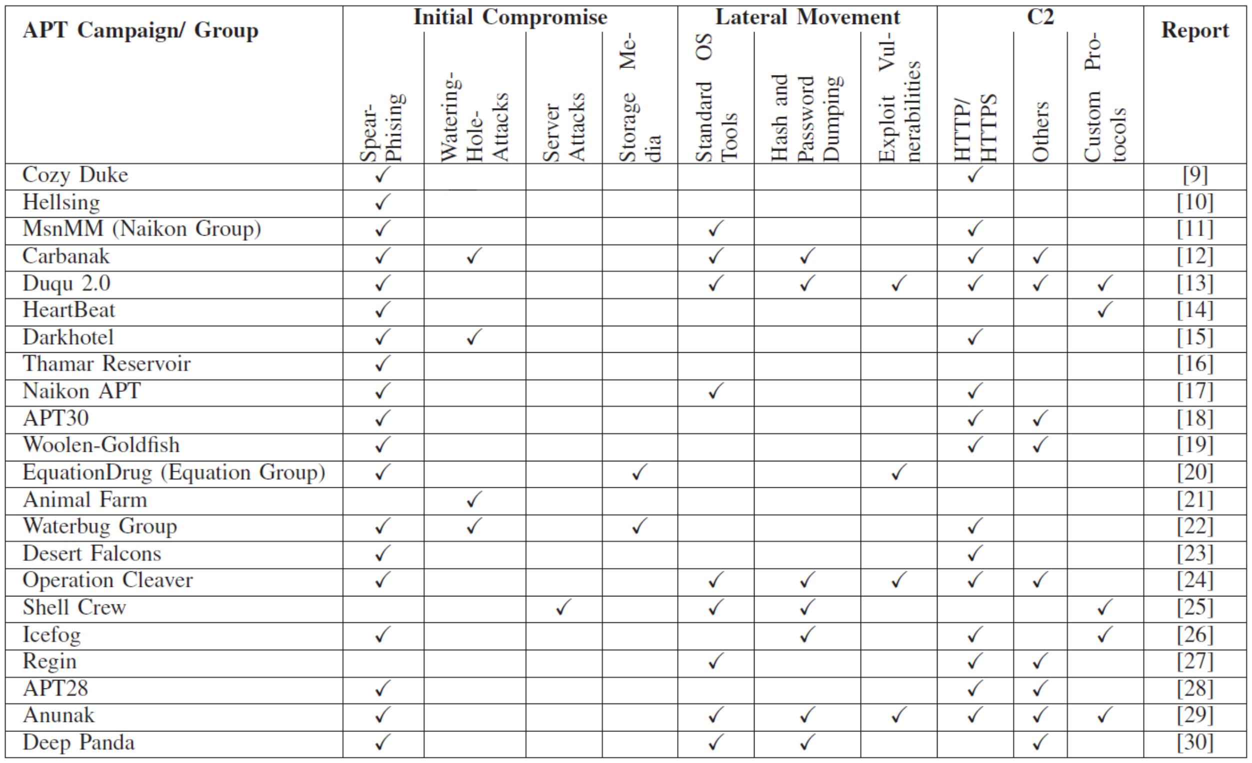
Ussath, M., Jaeger, D., Cheng, F., Meinel, C., & Institute, H. P. (2016). Advanced Persistent Threats: Behind the Scenes. *2016 Annual Conference on Information Science and Systems (CISS)*. 2016 Annual Conference on Information Science and Systems (CISS).

Verizon. (2020). *2020 Data Breach Investigation Report*. Verizon. https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf?mkt_tok=eyJpIjoiTlRjNE1ETTBZamN3WVRZMSIsInQiOiI4TEFaVXRHWm1kNmVhYnUrSkNCem90cjlKdnZURmI5QTM0N3c3c1RZUE5hcGRtbHhuakJkWFlzeWtjY1hqXC9wOVo1NXZEYkxoSENvWktuMmxEMTRKRlljNiswbmxNZWF1SmxVTFc4eEtsa1RQNTRnWEpkRVBpN05JQmpIQVB6bEYifQ%3D%3D

Weiner, R. (2018, September 21). Hacker linked to Target data breach gets 14 years in prison. *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

Yu, Y., Long, J., & Cai, Z. (2017). Network Intrusion Detection through Stacking Dilated Convolutional Auto encoders. *Security and Communication Networks*, *2017*, 1–10.https://doi.org/10.1155/2017/4184196

Appendix A



APT Campaign/ Group	Initial Compromise				Lateral Movement			C2			Report
	Spear-Phising	Watering-Hole-Attacks	Server Attacks	Storage Media	Standard OS Tools	Hash and Password Dumping	Exploit Vulnerabilities	HTTP/ HTTPS	Others	Custom Protocols	
Cozy Duke	✓							✓			[9]
Hellsing	✓										[10]
MsnMM (Naikon Group)	✓				✓			✓			[11]
Carbanak	✓	✓			✓	✓		✓	✓		[12]
Duqu 2.0	✓				✓	✓	✓	✓	✓	✓	[13]
HeartBeat	✓									✓	[14]
Darkhotel	✓	✓						✓			[15]
Thamar Reservoir	✓										[16]
Naikon APT	✓				✓			✓			[17]
APT30	✓							✓	✓		[18]
Woolen-Goldfish	✓							✓	✓		[19]
EquationDrug (Equation Group)	✓			✓			✓				[20]
Animal Farm		✓									[21]
Waterbug Group	✓	✓		✓				✓			[22]
Desert Falcons	✓							✓			[23]
Operation Cleaver	✓				✓	✓	✓	✓	✓		[24]
Shell Crew			✓		✓	✓				✓	[25]
Icefog	✓					✓		✓		✓	[26]
Regin					✓			✓	✓		[27]
APT28	✓							✓	✓		[28]
Anunak	✓				✓	✓	✓	✓	✓	✓	[29]
Deep Panda	✓				✓	✓			✓		[30]

(Ussath, Martin, et al. “Advanced Persistent Threats: Behind the Scenes.” 2016 Annual Conference on Information Science and Systems (CISS), Page 4, IEEE, 2016.)

Appendix B

Name:	Application Isolation and Sandboxing	Disable or Remove Feature or Program	Exploit Protection	Network Segmentation	Privileged Account Management	Threat Intelligence Program	Update Software	Vulnerability Scanning	Filter Network Traffic	Network Intrusion Prevention	User Account Management	Multi-Factor Authentication	Limit Hardware Installation	Active Directory Configuration	Passwor d Policies	Remote Data Storage	User Training	Execution Prevention	Restrict File and Directory Permissions
Exploitation of Remote Services	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Internal SpearPhishing	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Lateral Tool Transfer	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Remote Service Session Hijacking	🟥	🟩	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Remote Services	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Replication through Removeable Media	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥
Software Deployment Tools	🟥	🟥	🟥	🟩	🟩	🟥	🟩	🟥	🟥	🟥	🟩	🟩	🟥	🟩	🟩	🟩	🟩	🟥	🟥
Taint Shared Content	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩
Use Alternate Authentication Material	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

(MITRE ATT&CK Framework Recommended Remediations for Common Lateral Movement Attack Methods.)