Security Monitoring of Windows Containers

Peter Di Giorgio

Security Monitoring of Windows Containers

GIAC (GCIH) Gold Certification

Author: Peter Di Giorgio, peter.di.giorgio00@gmail.com
Advisor: *Tanya Baccam*

Accepted: *3/14/19*

Abstract

The information technology community has utilized container technology since the LXC project began in 2008 (Hildred, 2015). Containers are a form of virtualization that package application code and its dependencies together. Containers share the operating system kernel but maintain isolated processes. Until recently, it was not possible for the Windows operating system to share its kernel. As such, developers were long unable to package many Windows-specific applications into containers. However, after ten years of waiting, Microsoft finally delivered Windows containers in 2018. Today, container security best practices focus on container integrity and container host security. The industry is just beginning to consider techniques to monitor Windows containers. This research focuses on the possibility of using known techniques and open source tools to extract Windows event logs, processes, services, and registry data from containers to observe attacks.

1. Introduction

The information technology market is exploding. Companies and households invest millions of dollars annually acquiring, maintaining, and upgrading information systems. With so much invested in computing resources, many organizations are searching for creative ways to reduce the total cost of ownership for their technology. Containers are proving to be a pivotal technology for cost-effective software deployments. The ability to run custom software for an extended time is an effective financial practice, but it remains to be seen if companies can afford the security risk of running legacy code in modern enterprise environments.

Legacy systems are a common challenge in many enterprise networks. The cost and time of modernizing is difficult for even the most successful companies to balance. Container implementation is a critical step in legacy application modernization. There are security benefits to running legacy Windows applications on modern versions of the operating system. While container security best practices are developing, there aren't many existing discussions on how to monitor Windows containers for indications of an attack. With the global median dwell time for attacks in 2017 sitting at 101 days (FireEye, 2018), organizations need to consider more effective observation strategies for their environments.

The Windows operating system dominates the enterprise environment market share at 87% (Net Marketshare, 2018). Countless security professionals have studied the best techniques to monitor Windows systems and network traffic. These techniques may be equally effective at extracting Windows event logs, processes, services, and registry data from containers to observe attacks.

This paper will demonstrate the effectiveness of best practices for Windows environments to detect attacks within Windows containers. If well-known techniques and opensource tools prove to be effective, information security professionals can rapidly integrate those tools into their container environments.

2. Windows Containers

Microsoft and Docker established a strong partnership to make Windows containers a reality. Both companies provide extensive documentation on this new technology. While there are a lot of container implementation techniques, Docker's implementation and the Docker engine, purpose-built for the Windows operating system, will remain a constant for this research.

2.1. Windows Containers

Containers are a rapidly growing technology. To grasp container capabilities and limitations, an understanding of how containers work is required. Containers are an implementation of virtualization sometimes referred to as OS Virtualization (Russinovich, Containers: Docker, Windows, and Trends, 2015). Unlike more traditional forms of virtualization, containers do not depend on a significant amount of hardware virtualization. Containers share the operating system instead of owning the entire OS, like virtual machines do (Knulst, 2017). Docker employs a couple of techniques to deliver this revolutionary technology.

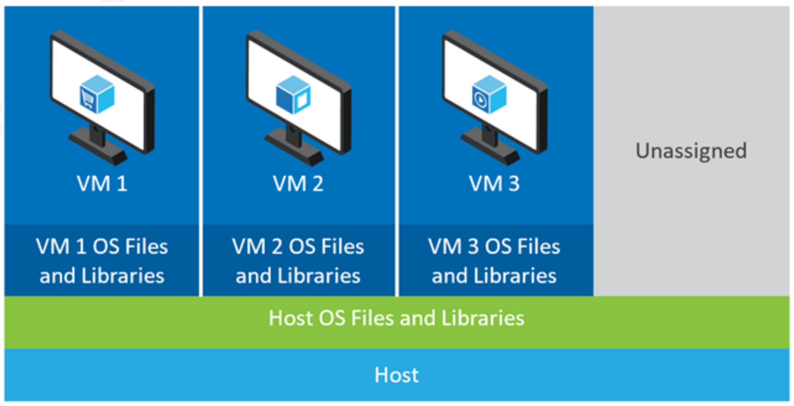

VM 1
VM 2
VM 3
Unassigned
VM 1 OS Files and Libraries
VM 2 OS Files and Libraries
VM 3 OS Files and Libraries
Host OS Files and Libraries
Host

Figure 1: Hardware Virtualization (Russinovich 2015)

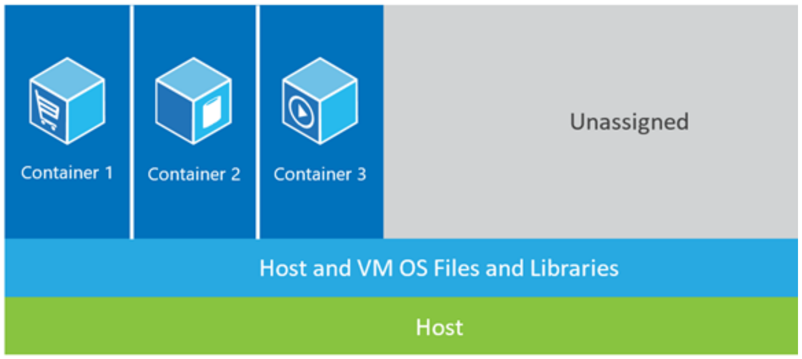


Figure 2: OS Virtualization (Russinovich 2015)

The first technique is namespace isolation which separates processes, networking activity, and filesystem behavior from the host operating system. Namespaces make it possible to have an isolated workspace that we know as a container (Docker, 2018). A container and its application receive the resources they are permitted to interact with, such as files, network ports, and running processes (Russinovich, Containers: Docker, Windows, and Trends, 2015). These are the only resources that the container can see on the host operating system. Namespaces are the reason that applications within a container look and feel like they are running in a clean operating system.

While it may seem like a container functions as a private operating system, many containers will share host operating system resources or files. This sharing is the foundation of container resource efficiency. The second technique used to maintain the illusion of a single operating system is Docker's "copy-on-write" method for shared resources. When an application makes changes to its container, like modifying a shared host operating system file, the container copies the file into the container (Russinovich, Containers: Docker, Windows, and Trends, 2015) before writing to it. This behavior ensures that the container keeps only what the application needs, and allows the container to move with everything the application needs to run.

Unlike virtual machines, a container does not need fixed allocations of resources for its processes. The host manages the resources, such as CPU, RAM, and network bandwidth, that a container can receive. Host management ensures that a container receives the resources it needs without impacting performance (Russinovich, Containers: Docker, Windows, and Trends, 2015) of other containers or applications running on the host system. Unless many applications all try to use a lot of CPU at the same time, dozens of containers will run concurrently on modest hardware (Stoneman, 2017). A container can have its resource limits set before starting, to restrict access to CPU or memory resources if necessary (Stoneman, 2017).

Namespace isolation, "copy-on-write," and host resource management make containers efficient, fast, and mobile. Containers are perfect for deploying a wide variety of applications. With the potential to run so many applications concurrently on-premises or in the cloud, it will become increasingly important to monitor their behavior for malicious activity.

Build
Develop an app using Docker containers with any language and any toolchain.
Ship
Ship the "Dockerized" app and dependencies anywhere - to QA, teammates, or the cloud - without breaking anything.
Run
Scale to 1000s of nodes, move between data centers and clouds, update with zero downtime and more.

Figure 3: Docker build-ship-run workflow (Knoll 2018)

2.2. Windows Enterprise Environments

A critical component of a Windows enterprise environment is a domain controller which provides identity and access control for the domain. Domain controllers are instances of the Windows server operating system running Active Directory Domain Services (ADDS) (Microsoft, 2014). Active Directory Domain Services provide hierarchical data storage for objects in a network such as users, computers, printers, and services (Microsoft, 2018). ADDS also provides a way to search for objects and read their properties. ADDS are central to the identity management, authentication, and administration of every node that operates within a Windows enterprise environment. Containers in a Windows environment can use group- managed service accounts as objects in Active Directory. These objects allow applications in containers to interact with ADDS as an authenticated service.

2.3. Observations of Events in the Windows Environment

Security architecture must address a couple of considerations before observing attacks in the Windows environment. First, the generation of events relevant to the threats must occur. Second, the event collection and transportation should happen. Finally, event correlation and analysis must occur to determine the impact of observed activity in the network. These considerations help in the selection of technology.

Cost, efficiency, and effectiveness are relevant factors in selecting capabilities to build a security monitoring ecosystem. Large companies can typically justify and afford expensive applications and appliances for security monitoring. Small businesses, nonprofits, or government entities may not have that luxury. Thankfully, there are open source solutions that offer efficiency and effectiveness. Windows event logs, Sysmon, and Wazuh all generate events with well-documented configurations. Event collection occurs through Wazuh or Winlogbeat from the Elastic Stack project. Finally, Security Onion offers an open source application stack to store, correlate, and analyze host and network events.

2.3.1. Windows Event Logs

There are a couple of ways to generate events in the Windows operating system. The operating system will generate log events and store them in Windows Event logs. A large number of services can generate logs. An obvious log source for security monitoring is the Security log of the Windows OS, but it isn't the only way to receive accurate indications of malicious activity. Here are some additional Windows event logs to collect (Lee & Pilkington, 2018):

- Application
- Security
- System
- Windows PowerShell
- Microsoft-Windows-PowerShell/Operational
- Microsoft-Windows-RemoteDesktopServices-RdpCoreTS/Operational
- Microsoft-Windows-SmbClient/Security
- Microsoft-Windows-SMBServer/Security
- Microsoft-Windows-TaskScheduler/Operational
- Microsoft-Windows-TerminalServices-RemoteConnectionManager/Operational
- Microsoft-Windows-Windows Defender/Operational
- Microsoft-Windows-Windows Firewall With Advanced Security/Firewall
- Microsoft-Windows-Winlogon/Operational
- Microsoft-Windows-WinRM/Operational
- Microsoft-Windows-WMI-Activity/Operational

2.3.2. System Monitor (Sysmon)

Another popular application that can be used to generate events is System Monitor (Sysmon). Sysmon is a system service developed by Microsoft as part of the Windows Sysinternals suite of tools. Sysmon monitors and logs system activity such as process creation, network connections, registry changes, and changes to file creation time (Russinovich & Garnier, Sysmon v8.04, 2017). Sysmon logs its events to a Windows event log named Microsoft-Windows-Sysmon/Operational. With a configuration file like the one at https://github.com/SwiftOnSecurity/sysmon-config, Sysmon has impressive capabilities (Russinovich & Garnier, Sysmon v8.04, 2017):

- Log process creation with the command line for both current and parent processes.
- Sysmon records the hash of process image files using SHA1 (the default), MD5, SHA256 or IMPHASH.
- Sysmon supports multiple hashes at the same time.
- Includes a process GUID in process creation events to allow for correlation of events even when Windows reuses process IDs.
- Includes a session GUID in each event to allow correlation of events for the same login session.
- Sysmon logs the loading of drivers or DLLs with their signatures and hashes.
- Logs open for raw read access of disks and volumes
- Optionally logs network connections, including each connection's source process, IP addresses, port numbers, hostnames and port names.
- Detects changes in file creation time to determine the actual creation of a file. Modification of file create timestamps is a technique commonly used by malware to cover its tracks.
- Automatically reloads configuration if changed in the registry.
- Rule filtering to include or exclude certain events dynamically.
- Generates events from early in the boot process to capture activity made by even sophisticated kernel-mode malware.

2.3.3. Wazuh

Wazuh is a security detection, visibility, and compliance open source project. It was born as a fork of OSSEC HIDS, and was later integrated with Elastic Stack and OpenSCAP, evolving into a more comprehensive solution (Wazuh, 2018). Of the many components of the project, the Wazuh agent can be used to monitor a variety of operating systems. Figure 4 outlines the agent's potential in endpoints:

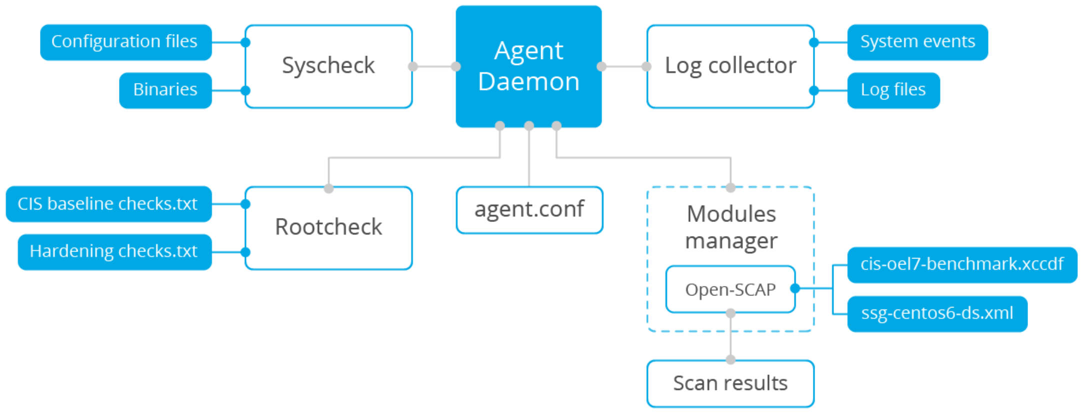


Figure 4: Wazuh Agent Capabilities (Wazuh, 2018)

2.3.4. Winlogbeat

Winlogbeat is a Windows log collection agent that was built for Elastic Stack. It ships event logs to Elasticsearch or Logstash and can run as a service (Elastic, 2019). Winlogbeat's configuration file is easily manipulated to send application, hardware, security, and system events to Elastic Stack.

2.3.5. Security Onion

Security Onion is a free and open source Linux distribution for intrusion detection, enterprise security monitoring, and log management (Security Onion, 2019). Security Onion brings full packet capture, network intrusion detection, host intrusion detection, and analysis tools together into one system (Security Onion, 2018). A variety of events generated by Windows, Sysmon, or Wazuh move to Security Onion for correlation with network data. Security Onion's SIEM becomes a one-stop-shop for analysis of a network and host-based events.

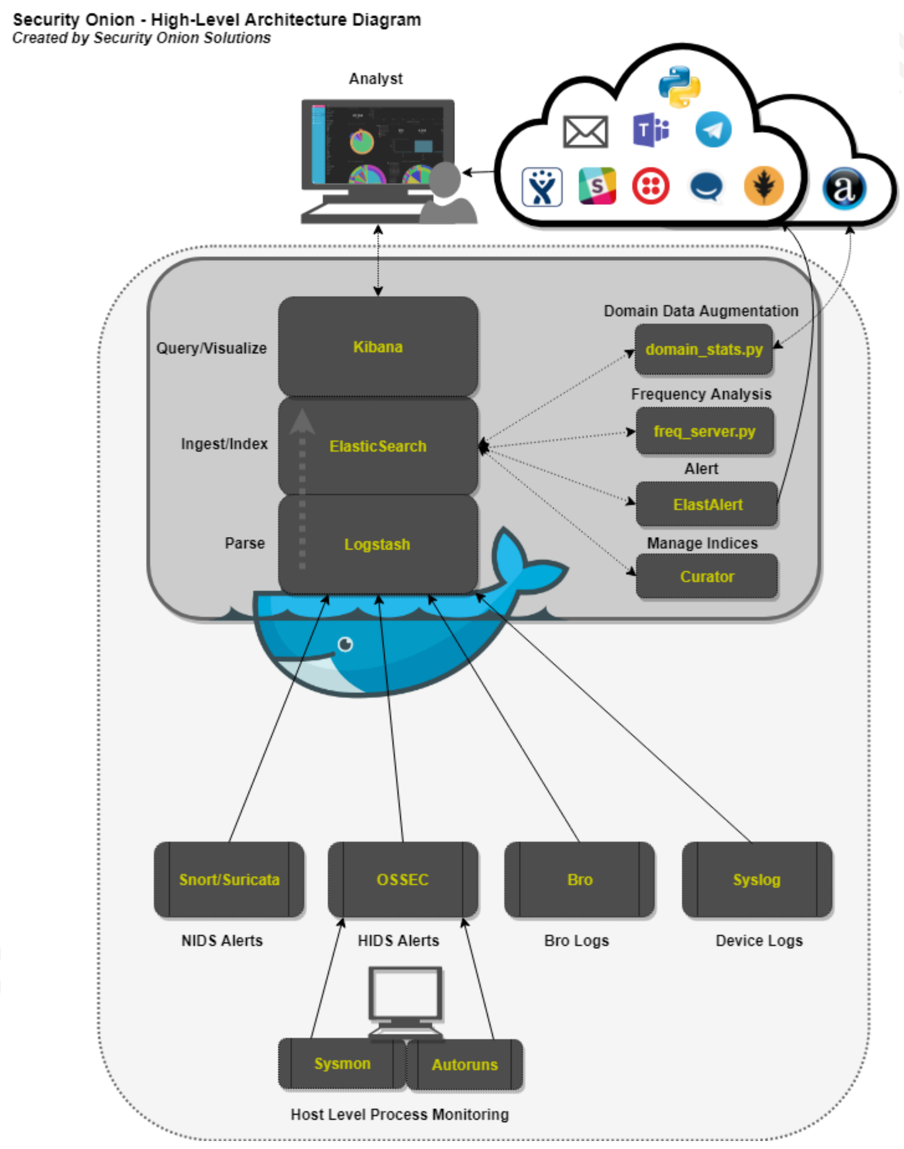


Figure 5: Security Onion High-Level Architecture (Security Onion, 2018)

3. Attacks Against Windows Containers

Several models characterize attacks against Windows Enterprise environments. One of the most comprehensive frameworks is MITRE's Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK). The first MITRE ATT&CK model was created in 2013 with a focus on the Windows enterprise environment (Strom, et al., 2018). This model is intended to be a mid-level abstraction of adversary behavior.

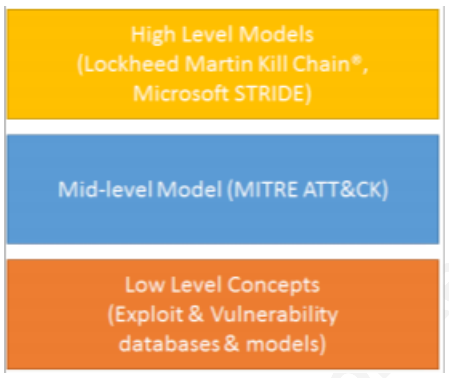

High Level Models (Lockheed Martin Kill Chain®, Microsoft STRIDE)
Mid-level Model (MITRE ATT&CK)
Low Level Concepts (Exploit & Vulnerability databases & models)

Figure 6: Abstraction Comparison (Strom et al., 2018)

The MITRE ATT&CK model nests with the Lockheed Martin Kill Chain. Introduced in 2011, the Lockheed Martin Kill Chain is a seven-stage model for computer network attack or exploitation. The stages are reconnaissance, weaponization, delivery, exploitation, installation, command and control (C2), and actions on the objective (Hutchins, Cloppert, & Amin, 2011). The ATT&CK model builds on the Lockheed Martin Kill Chain by categorizing adversary behavior into tactics and techniques under the seven stages.

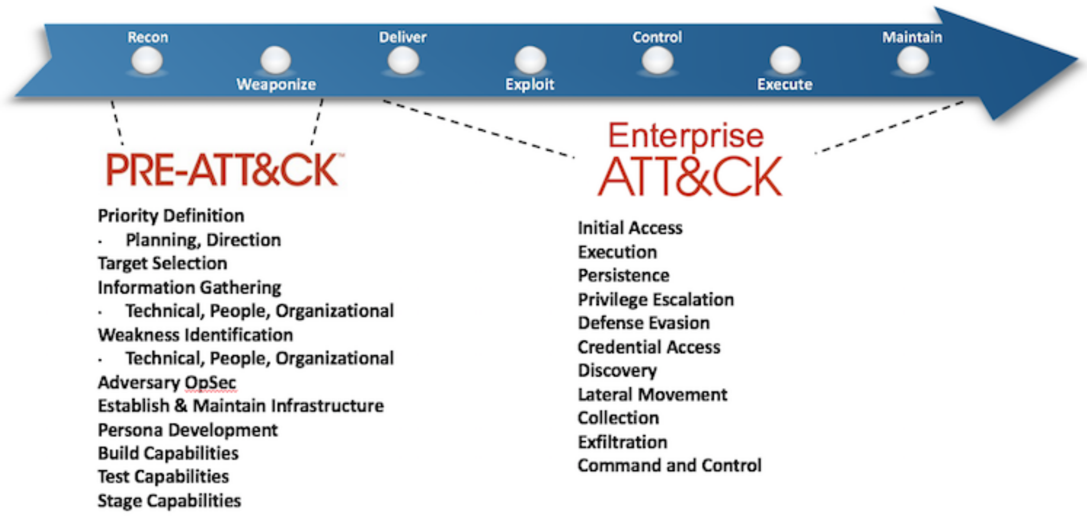


Figure 7: ATT&CK Enterprise Lifecycle (MITRE Corporation, 2018)

To demonstrate the security monitoring potential for Windows containers, the MITRE ATT&CK model will be used to plan an attack against a Windows environment. The penetration test campaign will begin by first accessing the .NET application, laterally moving through the small network, and will culminate with exfiltration of credentials from the domain controller. Using the MITRE ATT&K matrix, initial access techniques will include the exploitation of a public- facing application or the use of valid accounts. Execution will occur via command line, PowerShell, or service. Lateral movement will occur through Windows Remote Management or Windows admin shares. Kali will be used as the open source penetration test platform to leverage the Metasploit framework or PowerShell Empire.

3.1. Initial Access

The MITRE ATT&K model for enterprise networks catalogs ten techniques for gaining initial access (MITRE Corporation, 2018). Two techniques will be the focus of this particular research. Exploitation of a public-facing application and use of valid credentials are the most direct access vectors to a containerized application. The other eight techniques are important but provide indirect methods of access.

Public-facing applications are at high risk of exploitation. Vendors and security researchers are in a constant battle to find and fix vulnerabilities. In the last five years, the information technology community disclosed four vulnerabilities in Microsoft Internet Information Services (IIS) and twelve vulnerabilities in Microsoft SQL Server (MSSQL) (Ozkan & Keller, n.d.). While the number of vulnerabilities doesn't appear to be overly concerning, these examples are two of the most popular products used in public-facing web applications.

Credential stealing, or the use of stolen valid credentials during attacks, is one of the most dominate access vectors. In recent studies, approximately 81% of hijacking-related attacks leveraged stolen or weak credentials (Verizon, 2017). Additionally, 46% of web application breaches succeeded through the use of stolen credentials (Verizon, 2017). With time on their side, attackers use a variety of techniques to harvest valid credentials. Bruteforce, cracking, man-in-the-middle, and phishing are just a handful of techniques used to target users and administrators. Password reuse aggravates the problem. Valid credentials may not be the most sophisticated attack vector, but it is one of the most effective.

3.2. Execution

The MITRE ATT&K model offers over 30 techniques used by attackers in the execution phase. With so many options, it's not a surprise that the average dwell time of a compromise is 101 days (FireEye, 2018). The lab environment for this study is comprised of Windows virtual machines and Windows containers to create a small Windows enterprise. This environment is ideal for execution through the use of PowerShell, Windows Remote Management (WinRM), and the command prompt. All three components enable persistence, privilege escalations, and lateral movement, as well.

Windows Remote Management (WinRM) is a standard Simple Object Access Protocol (SOAP)-based, firewall-friendly protocol that allows hardware and operating systems, from different vendors, to interoperate (Microsoft, 2018). WinRM is a popular tool for Windows administrators, which makes it ideal for attackers. Until OpenSSH became available in Windows 10 and Server 2016, WinRM was the command line transportation tool of choice for administrators in the Windows environment. Its interoperability with a variety of Windows and third-party components make it an important part of any enterprise environment.

PowerShell and Command shell provide command-line and scripting capabilities to automate tasks in Windows environments. Command shell was the first shell built into the Windows operating system and is still available in the most current versions of Windows (Microsoft, 2018). PowerShell is a more robust shell which provides access to more than just simple operating system tasks. It allows administrators to access the registry and certificate stores as easily as they access the file system (Microsoft, 2018). PowerShell is a fully developed scripting language as well. When coupled with WinRM, PowerShell is a powerful tool used by administrators, or attackers, in a Windows enterprise.

3.3. Exfiltration

The ultimate goal of most breaches is data exfiltration. Whether it is intellectual property or bulk user data, attackers only profit from their activities if they can get information out of the network. Compression, encryption, and rate limiting are common in sophisticated attacks. This experiment will use a simple command and control system to harvest information from the network.

4. Findings and Discussion (Exposition of the Data)

The environment for this study remained simple to focus on Windows container security monitoring. The network architecture was flat to avoid complications created by hierarchical network connections between nodes. This network architecture provided a focus on the Windows events and pertinent network traffic achievable.

The lab environment is comprised of Windows virtual machines and Windows containers to create a small Windows enterprise. Each container will authenticate to the domain with group -managed service accounts. The containers will be on the same network as the domain controller. One container will host an ASP.NET application and a second will host SQL database to support the application. Since antivirus within containers is still in development, antivirus products are not a focus of in this research.

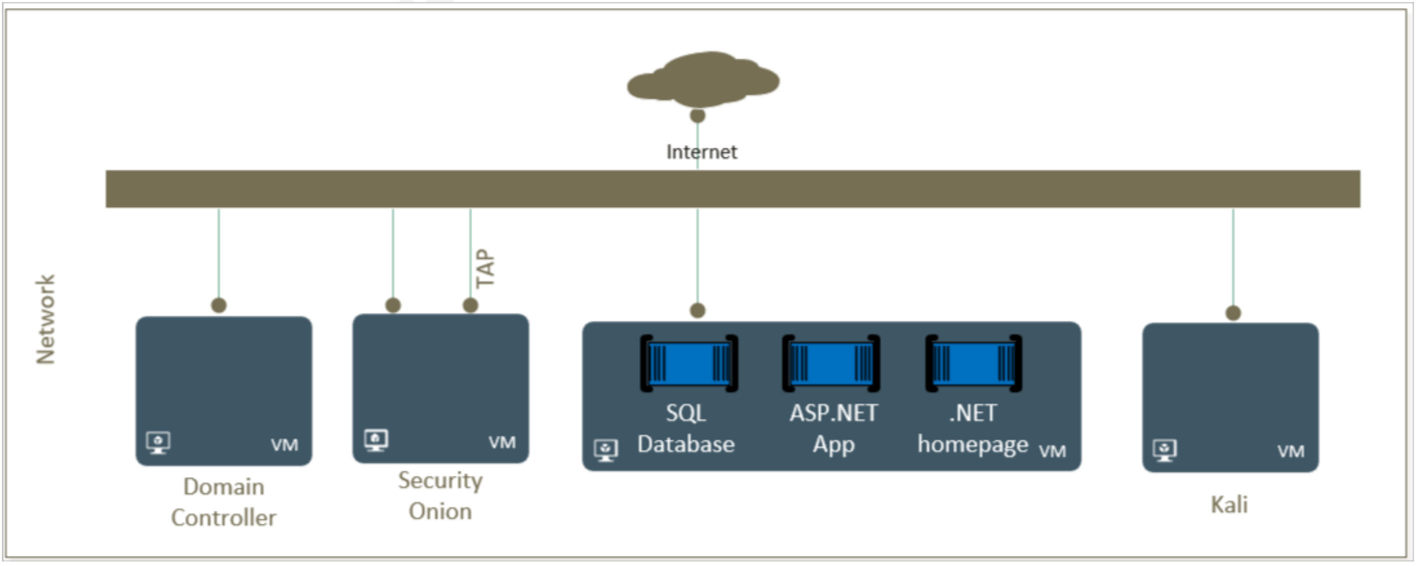
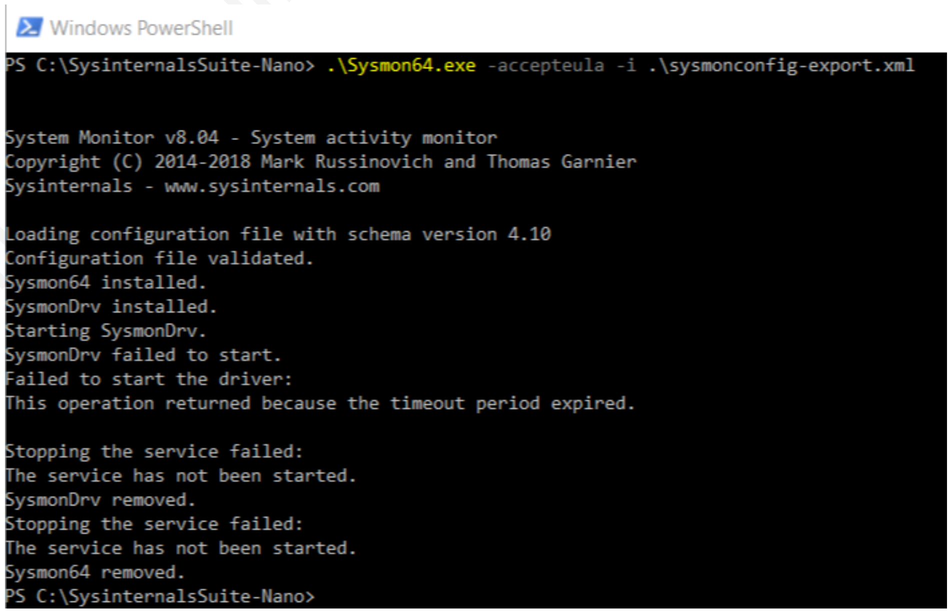


Figure 8: Container network

4.1. Event Generation and Collection from Windows Containers

The event generation and log collection had varied success. While the techniques selected may be low cost, their effectiveness and efficiency still must be determined. Sysmon did not successfully install in any of the containers which makes it ineffective. Wazuh was packaged as a Microsoft Installer (MSI) and required specific Windows components to install. The required Windows components are not available in every Windows container image so this option could bring inefficiencies to some production environments. Finally, Winlogbeat was the easiest to install in every container. Winlogbeat was also very easy to configure for the variety of containers tested. Each of these options can be beneficial in a production environment, but it's important to understand their strengths and weakness within Windows containers.

Sysmon resulted in most challenges. The installation script in Appendix B was successful on the Domain Controller, which was a Windows server core virtual machine. When the same script ran in the Windows containers, the Sysmon Driver (SysmonDrv) failed to start and the installation of the service aborted. The cause of the failure may be due to the namespace permissions used to create the container. After reviewing the logs extracted from the host and guest systems, namespace permissions or docker engine constraints appear to be the obstacle for the Sysmon Driver.



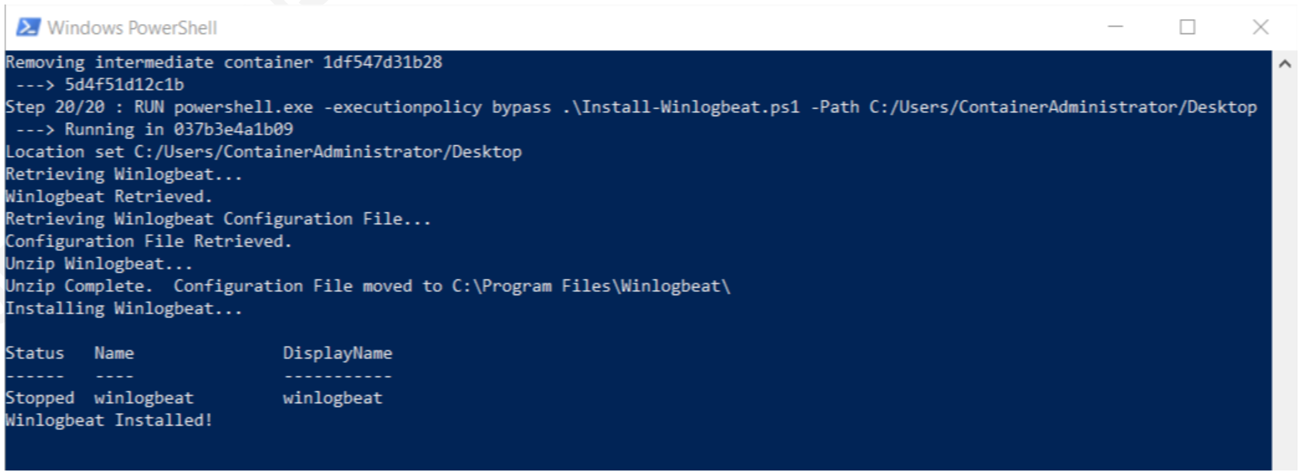
```
PS C:\SysinternalsSuite-Nano> .\Sysmon64.exe -accepteula -i .\sysmonconfig-export.xml


System Monitor v8.04 - System activity monitor
Copyright (C) 2014-2018 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.10
Configuration file validated.
Sysmon64 installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv failed to start.
Failed to start the driver:
This operation returned because the timeout period expired.

Stopping the service failed:
The service has not been started.
SysmonDrv removed.
Stopping the service failed:
The service has not been started.
Sysmon64 removed.
PS C:\SysinternalsSuite-Nano>
```

Figure 9: Sysmon Install Failed

The Wazuh agent installation was more complex than the other agents. The Windows container must have msiexec.exe to install the Wazuh agent. The Windows server core images have the required executable but don't always have the ASP.NET or MSSQL components for the applications. Since some images only contain the runtime of applications, application compilation cannot occur in every build. It may be necessary to publish the application then copy it into the Windows server core container with the runtime components for ASP.NET. Once the application is in the image, the Wazuh agent installation was possible.

Automated deployment of the Wazuh agent was an additional challenge. The installation from the script in Appendix C had a low success rate. Manual installation of the agent from the command line was the most effective. While the commands were identical, installation from the command line was successful but slower. The manual method also prevented integration of the installation into the image build process.

The automation of Winlogbeat was the most successful. The script in Appendix D completed successfully in the container build process. The agent also successfully reported to Logstash once the container ran. An additional benefit to the ease of installation was the integration with Elastic Stack in Security Onion. The logs were easy to read and find. Of the three methods, Winlogbeat proved to be the most efficient.

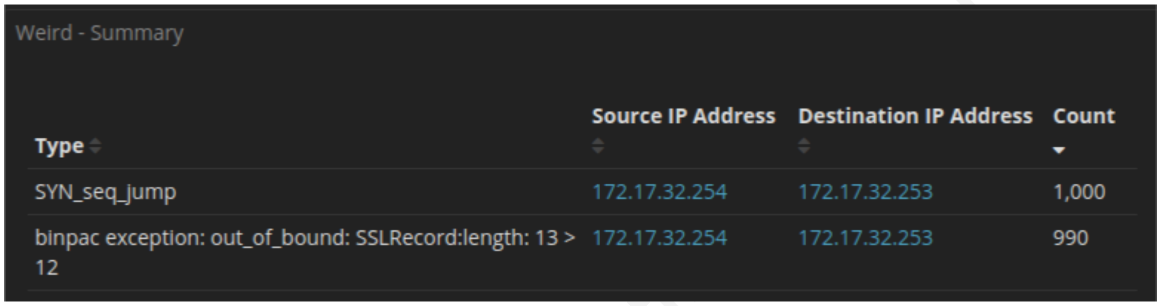


```
Removing intermediate container 1df547d31b28
 ---> 5d4f51d12c1b
Step 20/20 : RUN powershell.exe -executionpolicy bypass .\Install-Winlogbeat.ps1 -Path C:/Users/ContainerAdministrator/Desktop
 ---> Running in 037b3e4a1b09
Location set C:/Users/ContainerAdministrator/Desktop
Retrieving Winlogbeat...
Winlogbeat Retrieved.
Retrieving Winlogbeat Configuration File...
Configuration File Retrieved.
Unzip Winlogbeat...
Unzip Complete.  Configuration File moved to C:\Program Files\Winlogbeat\
Installing Winlogbeat...

Status   Name               DisplayName
------   ----               -----------
Stopped  winlogbeat         winlogbeat
Winlogbeat Installed!
```

Figure 10: Winlogbeat Install during image build

4.2. Observation of Attacks

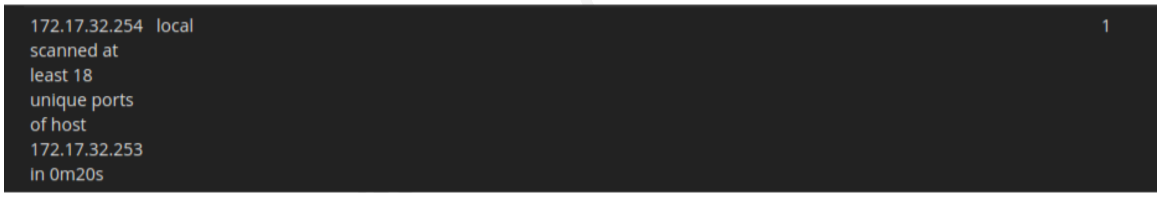
With a variety of monitoring techniques running in the environment, most of the attacks were visible from Security Onion's Elastic Stack. Security Onion served two functions in this test. Frist, Security Onion provided the network traffic monitoring through packet capture, Bro logs, and Snort signatures. Second, Security Onion can aggregate host logs and enable analysis through Elastic Stack. In such a simple network, the reconnaissance phase of the attack was not as difficult to observe as it would be in a production environment. Security Onion visualizations highlight small bursts of syn scans identified by Bro (now called Zeek).



Weird - Summary

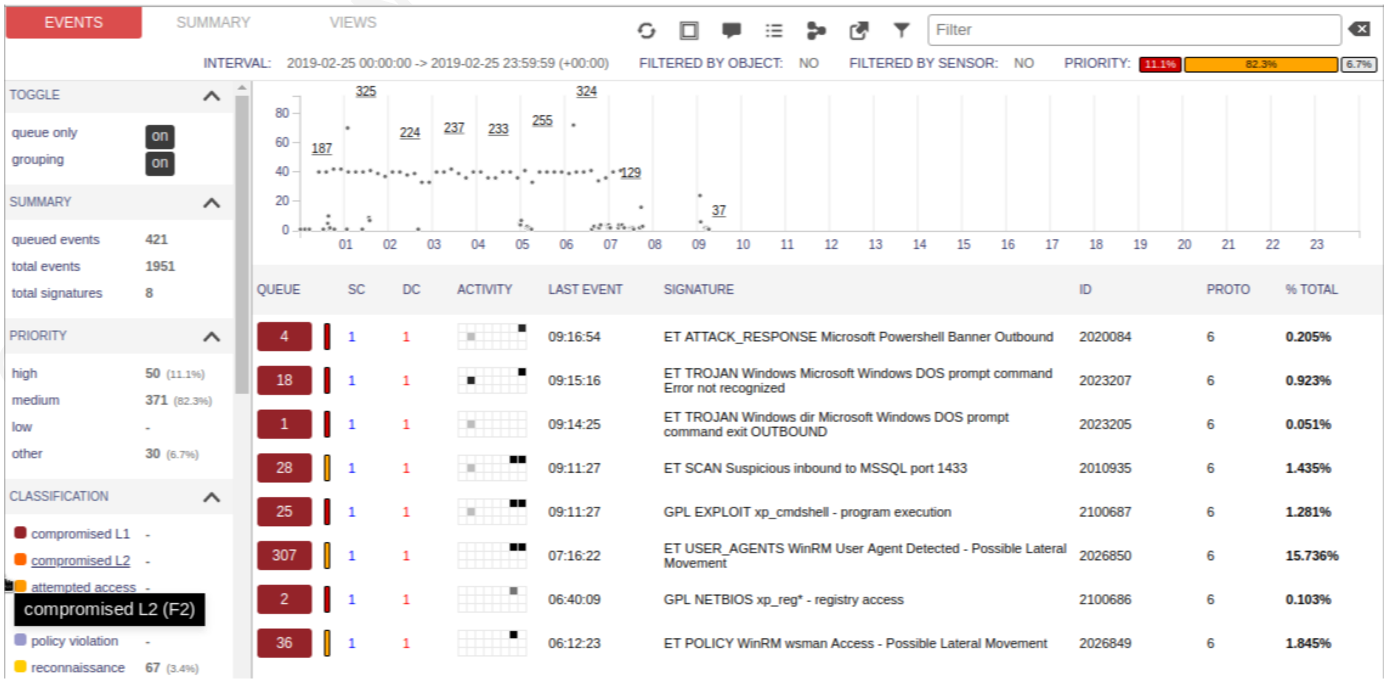
Type	Source IP Address	Destination IP Address	Count
SYN_seq_jump	172.17.32.254	172.17.32.253	1,000
binpac exception: out_of_bound: SSLRecord:length: 13 > 12	172.17.32.254	172.17.32.253	990

Figure 11: Bro Weird Log identifies SYN anomalies



```
172.17.32.254  local                                                   1
scanned at
least 18
unique ports
of host
172.17.32.253
in 0m20s
```

Figure 12: Snort logs from MSSQL abuse



INTERVAL: 2019-02-25 00:00:00 -> 2019-02-25 23:59:59 (+00:00) FILTERED BY OBJECT: NO FILTERED BY SENSOR: NO PRIORITY: 11.1% 82.3% 6.7%

TOGGLE

queue only	on
grouping	on

SUMMARY

queued events	421
total events	1951
total signatures	8

PRIORITY

high	50 (11.1%)
medium	371 (82.3%)
low	-
other	30 (6.7%)

CLASSIFICATION

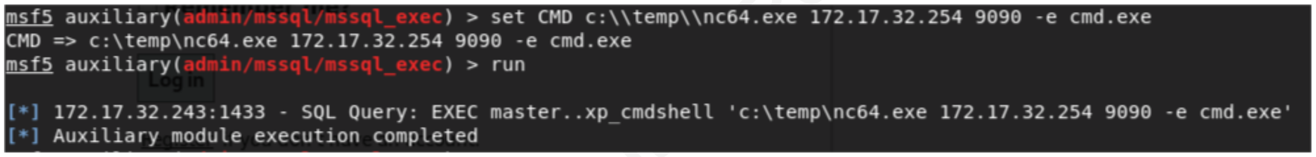
compromised L1	-
compromised L2	-
attempted access	-
policy violation	-
reconnaissance	67 (3.4%)

QUEUE	SC	DC	ACTIVITY	LAST EVENT	SIGNATURE	ID	PROTO	% TOTAL
4	1	1		09:16:54	ET ATTACK_RESPONSE Microsoft Powershell Banner Outbound	2020084	6	0.205%
18	1	1		09:15:16	ET TROJAN Windows Microsoft Windows DOS prompt command Error not recognized	2023207	6	0.923%
1	1	1		09:14:25	ET TROJAN Windows dir Microsoft Windows DOS prompt command exit OUTBOUND	2023205	6	0.051%
28	1	1		09:11:27	ET SCAN Suspicious inbound to MSSQL port 1433	2010935	6	1.435%
25	1	1		09:11:27	GPL EXPLOIT xp_cmdshell - program execution	2100687	6	1.281%
307	1	1		07:16:22	ET USER_AGENTS WinRM User Agent Detected - Possible Lateral Movement	2026850	6	15.736%
2	1	1		06:40:09	GPL NETBIOS xp_reg* - registry access	2100686	6	0.103%
36	1	1		06:12:23	ET POLICY WinRM wsman Access - Possible Lateral Movement	2026849	6	1.845%

Figure 13: Attacker identified conducting a port scan

As the attack progressed, the MSSQL database became the target. The ASP.NET 4.5 web application used in this environment did not offer an attack surface through input validation vulnerability or SQL Injection, but it did pass credentials in plain text.

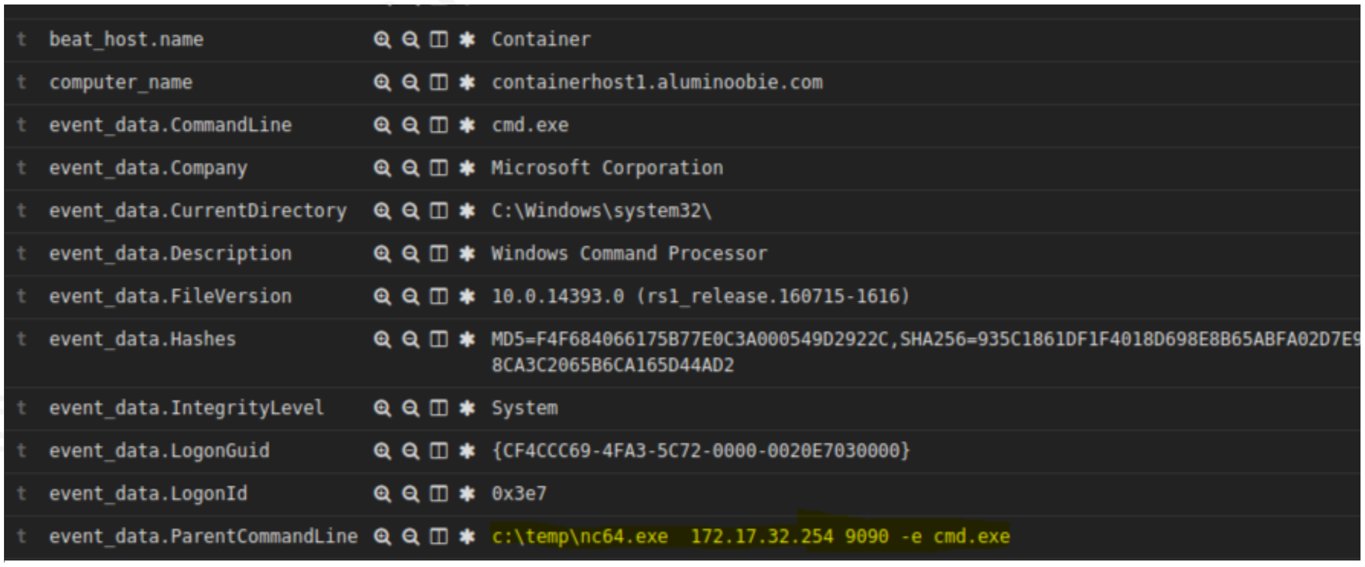
Credential reuse in web applications ultimately gave the attacker an avenue of approach. The attacker used the credentials to execute commands through MSSQL. The attacker used netcat compiled for Windows to send a command shell back to the attacker's system.



```
msf5 auxiliary(admin/mssql/mssql_exec) > set CMD c:\\temp\\nc64.exe 172.17.32.254 9090 -e cmd.exe
CMD => c:\temp\nc64.exe 172.17.32.254 9090 -e cmd.exe
msf5 auxiliary(admin/mssql/mssql_exec) > run

[*] 172.17.32.243:1433 - SQL Query: EXEC master..xp_cmdshell 'c:\temp\nc64.exe 172.17.32.254 9090 -e cmd.exe'
[*] Auxiliary module execution completed
```

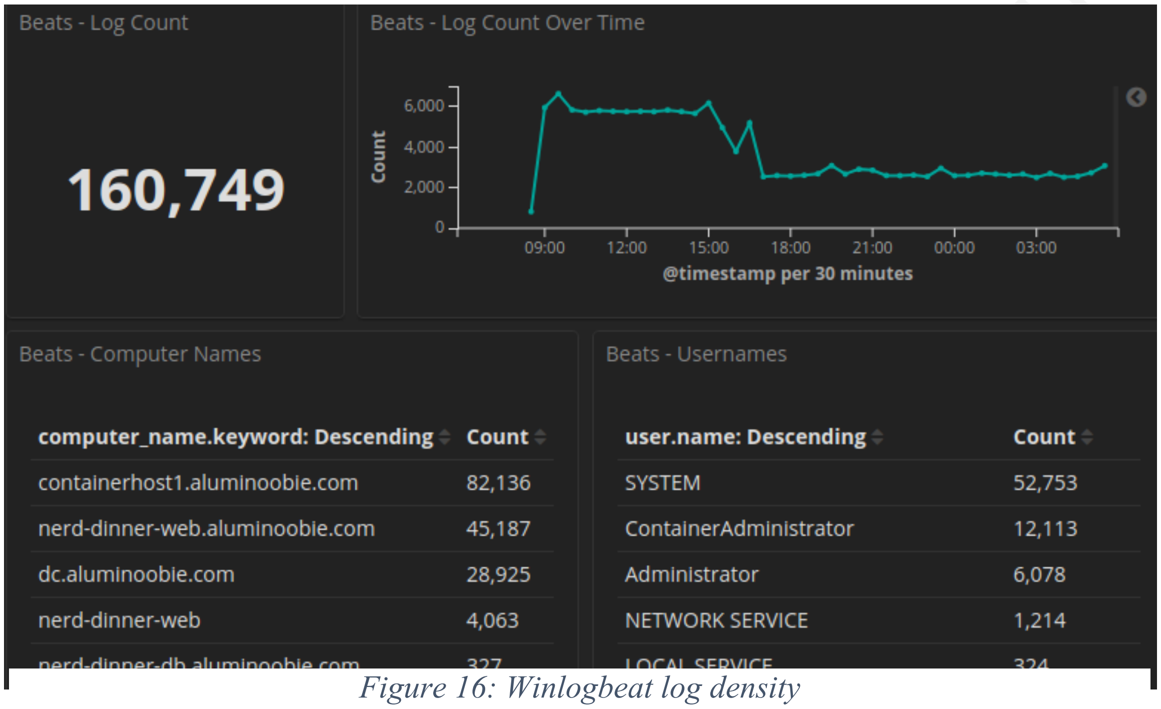
Figure 14: Shoveling a shell from MSSQL

The first indication of an attack came in the network intrusion detection logs. Snort with ET Open rules caught the basic attack. With so much information available, it was not difficult following the attack. Logs collected by Winlogbeat on the database container identified the netcat executable pushing the command shell.



```
t  beat_host.name               Container
t  computer_name                containerhost1.aluminoobie.com
t  event_data.CommandLine       cmd.exe
t  event_data.Company           Microsoft Corporation
t  event_data.CurrentDirectory  C:\Windows\system32\
t  event_data.Description       Windows Command Processor
t  event_data.FileVersion       10.0.14393.0 (rs1_release.160715-1616)
t  event_data.Hashes            MD5=F4F684066175B77E0C3A000549D2922C,SHA256=935C1861DF1F4018D698E8B65ABFA02D7E9
                                8CA3C2065B6CA165D44AD2
t  event_data.IntegrityLevel    System
t  event_data.LogonGuid         {CF4CCC69-4FA3-5C72-0000-0020E7030000}
t  event_data.LogonId           0x3e7
t  event_data.ParentCommandLine c:\temp\nc64.exe  172.17.32.254 9090 -e cmd.exe
```

Figure 15: Caught shoveling



Beats - Computer Names

computer_name.keyword: Descending	Count
containerhost1.aluminoobie.com	82,136
nerd-dinner-web.aluminoobie.com	45,187
dc.aluminoobie.com	28,925
nerd-dinner-web	4,063
nerd-dinner-db.aluminoobie.com	327

Beats - Usernames

user.name: Descending	Count
SYSTEM	52,753
ContainerAdministrator	12,113
Administrator	6,078
NETWORK SERVICE	1,214
LOCAL SERVICE	324

Figure 16: Winlogbeat log density

While it may seem like a lot of logs to collect, the collected logs were purposely selected based on lessons learned. Frameworks like MITRE ATT&K supported by breach reports from vendors can help security analysts accelerate the identification of an attack. With an initial indication of suspicious activity, the analyst has all the data to quickly carve through over one-hundred thousand logs to get down the several hundred relevant to the attack.

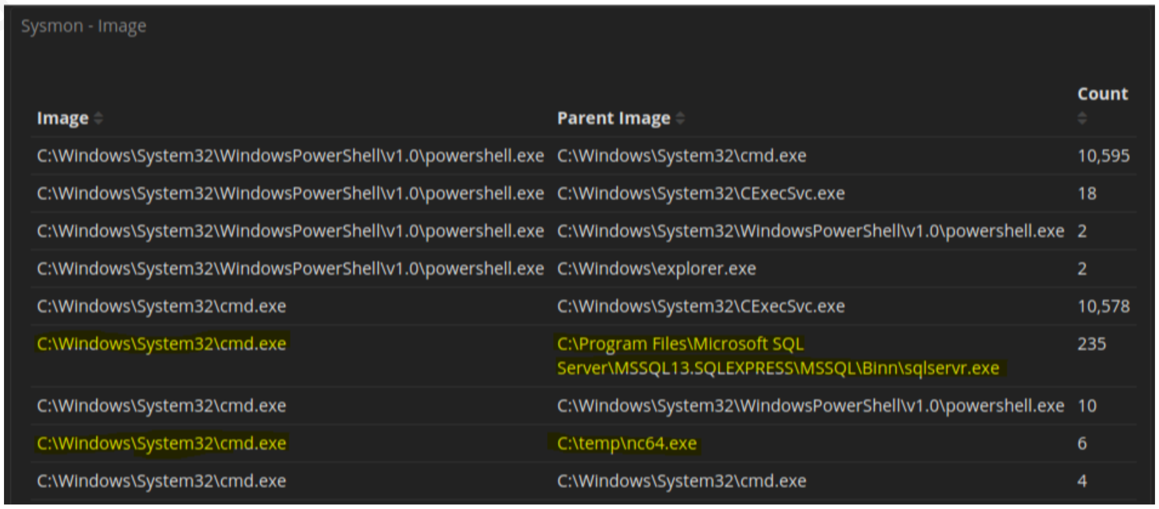
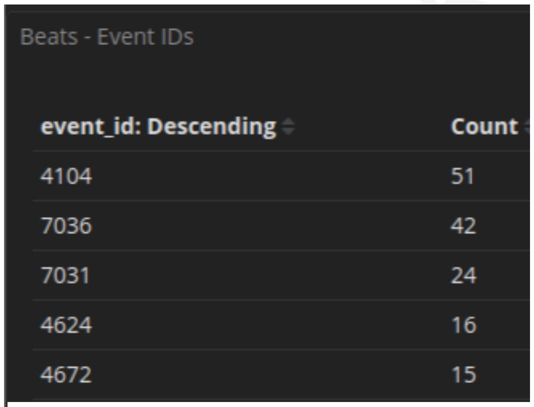

Sysmon - Image

Image	Parent Image	Count
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\cmd.exe	10,595
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\CExecSvc.exe	18
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	2
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\explorer.exe	2
C:\Windows\System32\cmd.exe	C:\Windows\System32\CExecSvc.exe	10,578
C:\Windows\System32\cmd.exe	C:\Program Files\Microsoft SQL Server\MSSQL13.SQLEXPRESS\MSSQL\Binn\sqlservr.exe	235
C:\Windows\System32\cmd.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	10
C:\Windows\System32\cmd.exe	C:\temp\nc64.exe	6
C:\Windows\System32\cmd.exe	C:\Windows\System32\cmd.exe	4

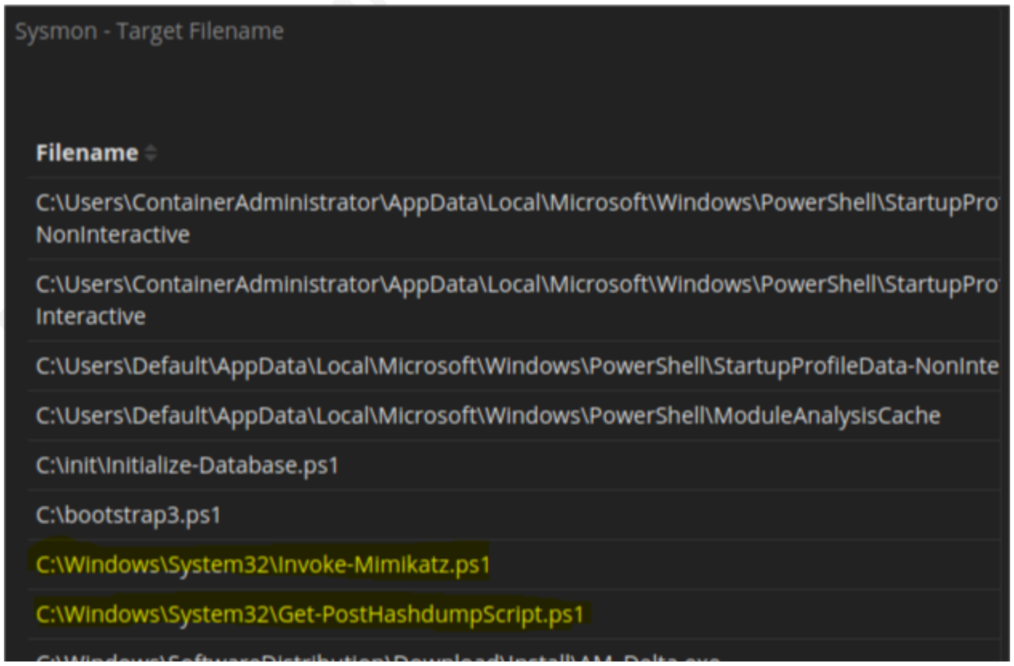
Figure 17: Suspicious Images

The event IDs in Figure 18 are unexpected in the containers for a web application. When PowerShell observes Script Blocks or suspicious scripts, Windows logs an Event ID 4104. Event ID 7036 indicates that a service started or stopped, which is concerning in the context of the remaining events. Event ID 4624 is a type 3 (remote) login, and Event ID 4672 means a login with elevated or administrative privileges. Figure 19 identifies the techniques used by the attacker to extract credentials from the container. All of this activity on a Windows container is a recipe for disaster, especially if a security operations center never collects these activities.


Beats - Event IDs

event_id: Descending	Count
4104	51
7036	42
7031	24
4624	16
4672	15

Figure 18: Filtered on one container


Sysmon - Target Filename

Filename
C:\Users\ContainerAdministrator\AppData\Local\Microsoft\Windows\PowerShell\StartupPro NonInteractive
C:\Users\ContainerAdministrator\AppData\Local\Microsoft\Windows\PowerShell\StartupPro Interactive
C:\Users\Default\AppData\Local\Microsoft\Windows\PowerShell\StartupProfileData-NonInte
C:\Users\Default\AppData\Local\Microsoft\Windows\PowerShell\ModuleAnalysisCache
C:\init\Initialize-Database.ps1
C:\bootstrap3.ps1
C:\Windows\System32\Invoke-Mimikatz.ps1
C:\Windows\System32\Get-PostHashdumpScript.ps1

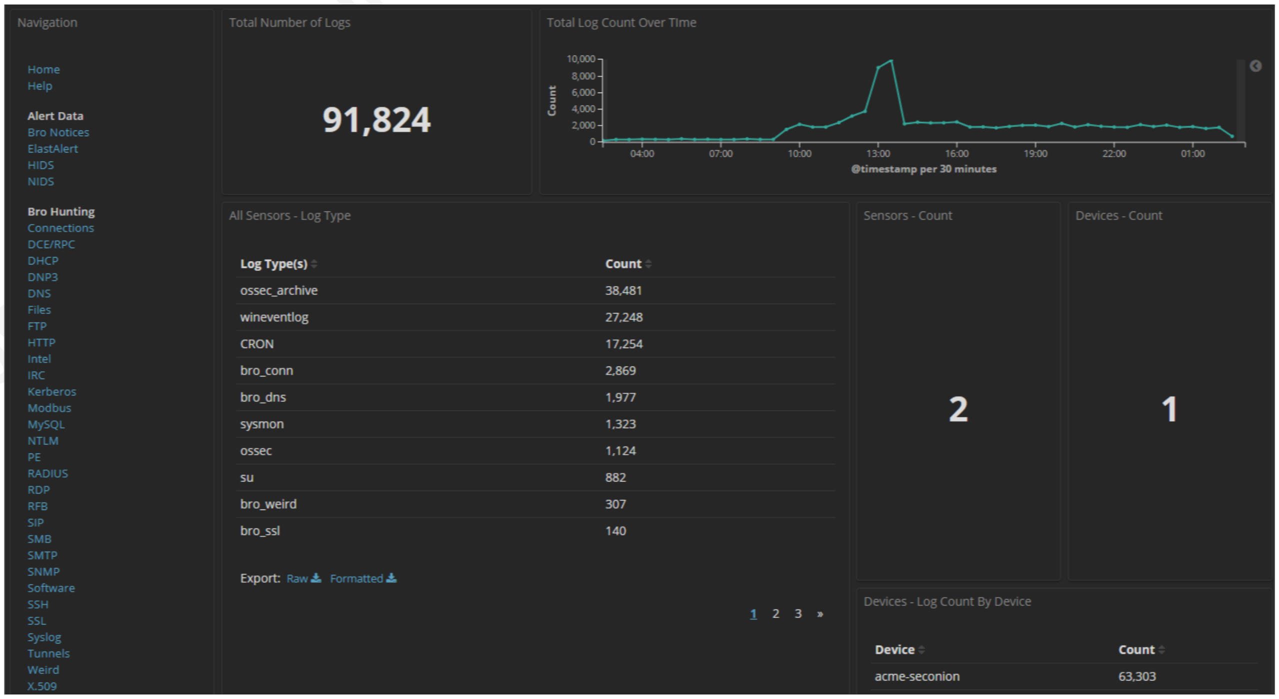
Figure 19: It's getting worse!

5. Recommendations and Implications

These findings scratch the surface of security monitoring in Windows containers. Windows container technology and images are evolving rapidly. Researchers, developers, and security professionals will continue to push the envelope of possibility in this space. It is unlikely that the findings of this paper will remain valid for too long. As such, it is important that additional work is done regarding the integration of security monitoring capabilities into container images.

5.1. Recommendations for Use in Production

The use of Winlogbeat as a log collection tool is a great option when coupled with Elastic Stack. The ability to bring endpoint events into Security Onion to correlate with network events proved to be a benefit for two reasons. First, the network events bring context and accelerate the identification of some attacks. Second, the Sysmon parsers for Security Onion capture appropriate logs from Winlogbeat and present them as if Sysmon collected the logs. Ultimately, the key to Winlogbeat's success in this research was its ease of installation during the Docker image build. Information security teams can provide their developers with a simple PowerShell script to include in the Dockerfile so that every container goes into production as a sensor, which reduces blind spots in an organization's enterprise monitoring architecture



Total Number of Logs

91,824

All Sensors - Log Type

Log Type(s)	Count
ossec_archive	38,481
wineventlog	27,248
CRON	17,254
bro_conn	2,869
bro_dns	1,977
sysmon	1,323
ossec	1,124
su	882
bro_weird	307
bro_ssl	140

Sensors - Count

2

Devices - Count

1

Devices - Log Count By Device

Device	Count
acme-seconion	63,303

Figure 20: Security Onion Kibana Dashboard with network and host events

5.2. Implications for the Future

Future research and development should address several areas. The first is testing in Windows Server or Azure. Windows 10 Professional was the environment for this research and was sufficient, but Windows Server or Azure could offer additional capability in virtual networking.

This research did not leverage Docker compose or other orchestration technologies. A more robust test environment would demonstrate the security challenges of Windows containers at scale.

Sysmon is a powerful tool in the Windows environment. The data collected during this research will be sent to the SysInternals team to inform improvements to Sysmon or the Docker Engine for Windows. Sysmon's ability to run in a Windows container significantly increases security teams' visibility of application and process behavior within the container.

6. Conclusion

There are low-cost options for monitoring Windows operating systems and containers. The techniques and technology exist to efficiently and effectively integrate security monitoring into a DevOps environment driven by container technology. Finally, there is potential for developers, administrators, and information security personnel to work together to secure their environments.

Windows containers are revolutionizing application and content delivery from the cloud. With so much potential for legacy applications to get new life in the cloud, the information security community must follow the technology and extend its visibility. As networks become more complex, dynamic observation techniques need to be integrated into development to ensure security of a Windows environment.

References

Docker. (2018, December 21). *Docker Overview*. Retrieved from Docker Docs: https://docs.docker.com/engine/docker-overview/

Elastic. (2019). *Winlogbeat Overview*. Retrieved from Elastic Docs: https://www.elastic.co/guide/en/beats/winlogbeat/current/_winlogbeat_overview.html#_winlogbeat_overview

FireEye. (2018). *M-Trends 2018 Report.* FireEye. Retrieved from https://www.fireeye.com/content/dam/collateral/en/mtrends-2018.pdf

Hildred, T. (2015, August 28). *The History of Conatiners*. Retrieved from Red Hat Enterprise Linux Blog: https://rhelblog.redhat.com/2015/08/28/the-history-of-containers/

Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains.* Lockheed Martin Corporation. Retrieved February 2, 2019, from https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

Knulst, C. (2017, February 28). Windows Containers: What is it and Why Should We Care? *SDN Magazine*(131), pp. 23-26.

Lee, R., & Pilkington, M. (2018, December). Hunt Evil Poster. *Poster_FOR508_V4.2_12-18*. SANS DFIR.

Microsoft. (2014, November 18). *Domain Controller Roles*. Retrieved from Docs: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc786438(v=ws.10)

Microsoft. (2018, May 30). *Active Directory Domain Services*. Retrieved from Windows Dev Center: https://docs.microsoft.com/en-us/windows/desktop/ad/active-directory-domain-services

Microsoft. (2018, August 26). *PowerShell*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-6

Microsoft. (2018, May 21). *Windows Command*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands

Microsoft. (2018, May 30). *Windows Remote Management*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/desktop/winrm/portal

MITRE Corporation. (2018). *ATT&CK for Enterprise*. Retrieved from MITRE ATT&CK: https://attack.mitre.org/resources/enterprise-introduction/

Net Marketshare. (2018, December). Operating System Market Share. Retrieved from https://netmarketshare.com/operating-system-market-share.aspx

Ozkan, S., & Keller, C. (n.d.). *Vulnerability Statistics*. Retrieved February 24, 2019, from CVE Details: https://www.cvedetails.com/product/251/Microsoft-Sql-Server.html?vendor_id=26

Russinovich, M. (2015, August 17). *Containers: Docker, Windows, and Trends*. Retrieved from Microsoft Azure: https://azure.microsoft.com/en-us/blog/containers-docker-windows-and-trends/

Russinovich, M., & Garnier, T. (2017, May 21). *Sysmon v8.04*. Retrieved from Sysinternals: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

Security Onion. (2018, September 12). *Introduction to Security Onion*. Retrieved from Github: https://github.com/Security-Onion-Solutions/security-onion/wiki/IntroductionToSecurityOnion

Security Onion. (2019). Retrieved from Security Onion Blog: https://blog.securityonion.net/

Stoneman, E. (2017). *Docker on Windows.* Birmingham: Packt Publishing Ltd.

Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018). *MITRE ATT&CK: Design and Philosophy.* McLean, VA: MITRE Corporation. Retrieved from https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf

Verizon. (2017). *2017 Data Breach Investigations Report.* New York City: Verizon. Retrieved from http://www.verizonenterprise.com/verizon-insights-lab/data-breach-digest/2017/

Wazuh. (2018, December 12). *Getting Started*. Retrieved from Wazuh Documentation: https://documentation.wazuh.com/current/getting-started/index.html

Figures

Appendix A: Example Dockerfile

```
# escape=`
FROM sixeyed/msbuild:netfx-4.5.2-ssdt AS builder

WORKDIR C:\src\NerdDinner.Database
COPY src\NerdDinner.Database .
RUN msbuild NerdDinner.Database.sqlproj `

/p:SQLDBExtensionsRefPath="C:\Microsoft.Data.Tools.Msbuild.10.0.61026\l
ib\net40" `

/p:SqlServerRedistPath="C:\Microsoft.Data.Tools.Msbuild.10.0.61026\lib\
net40"

# db image
FROM microsoft/mssql-server-windows-express

ENV ACCEPT_EULA="Y" `
    DATA_PATH="C:\data" `
    sa_password="N3rdD!Nne720^6"

VOLUME ${DATA_PATH}
WORKDIR C:\init

COPY Initialize-Database.ps1 .
CMD powershell ./Initialize-Database.ps1 -sa_password $env:sa_password
-data_path $env:data_path -Verbose
COPY ./Install-Sysmon.ps1 ./
COPY ./Install-Wazuh.ps1 ./
COPY ./Install-Winlogbeat.ps1 ./
RUN powershell.exe -executionpolicy bypass .\Install-Sysmon.ps1 `
      -Path C:/Users/ContainerAdministrator/Desktop
RUN powershell.exe -executionpolicy bypass .\Install-Wazuh.ps1 `
      -Path C:/Users/ContainerAdministrator/Desktop -agentname moviemvc
RUN powershell.exe -executionpolicy bypass .\Install-Winlogbeat.ps1 `
      -Path C:/Users/ContainerAdministrator/Desktop

COPY --from=builder
C:\src\NerdDinner.Database\bin\Debug\NerdDinner.Database.dacpac .
```

Appendix B: Install-Sysmon.ps1

```
<#
.SYNOPSIS
Install-Sysmon downloads SysInternals Suite and installs Sysmon
with a configuration file.
.DESCRIPTION
PowerShell script or module to install Sysmon with configuration
.PARAMETER path
The path to the working directory.  The default is user Documents.
.EXAMPLE
Install-Sysmon -path C:\Users\example\Desktop
#>

[CmdletBinding()]

#Establish parameters for path
param (
    [string]$path=[Environment]::GetFolderPath("Desktop")
)

#Test path and create it if required

If(!(test-path $path))
{
      Write-Information -MessageData "Path does not exist.  Creating
Path..." `
            -InformationAction Continue;
      New-Item -ItemType Directory -Force -Path $path | Out-Null;
      Write-Information -MessageData "...Complete" `
            -InformationAction Continue
}

[Net.ServicePointManager]::SecurityProtocol = `
      [Net.SecurityProtocolType]::Tls12

Set-Location $path

Write-Host "Location set $path"

Write-Host "Retrieving SysInternals Suite for Nano Server..."

Invoke-WebRequest `
      -Uri https://download.sysinternals.com/files/SysinternalsSuite-
Nano.zip `
      -Outfile SysinternalsSuite-Nano.zip

Write-Host "SysInternals Retrieved"

Write-Host "Unzip SysInternals..."

Expand-Archive SysinternalsSuite-Nano.zip

Set-Location $path\SysinternalsSuite-Nano

Write-Host "Unzip Complete."
```

```
Write-Host "Retrieving Configuration File..."

Invoke-WebRequest `
      -Uri https://raw.githubusercontent.com/aluminoobie/sysmon-
config/master/sysmonconfig-export.xml `
      -Outfile sysmonconfig-export.xml

Write-Host "Configuration File Retrieved."

Write-Host "Installing Sysmon..."

.\sysmon64.exe -accepteula -i sysmonconfig-export.xml

Write-Host "Sysmon Installed!"
```

Appendix C: Install-Wazuh.ps1

```
<#
.SYNOPSIS
Install-Wazuh downloads Wazuh Agent and installs Wazuh
with a configuration file.
.DESCRIPTION
PowerShell script or module to install Wazuh with configuration

Ensure that the ossec-authd daemon is running in the foreground
on your ossec server.

Linux command to find running ossec-authd processes on a server.
ps aux | grep ossec-authd | grep -v grep

Linux command to kill running ossec-authd processes
pkill ossec-authd

Linux command to run ossec-authd in foreground
./var/ossec/bin/ossec-authd -f -P

authd password is set in var/ossec/etc/authd.pass
echo ossec>/var/ossec/etc/authd.pass

.PARAMETER path
The path to the working directory.  The default is user Documents.
.PARAMETER agentname
The name of the agent that you are installing.  This parameter is the
agent name registered with the ossec server
.EXAMPLE
Install-Wazuh -path C:\Users\example\Desktop -agentname 'example'
#>

[CmdletBinding()]

#Establish parameters
param (
    [string]$path=[Environment]::GetFolderPath("Desktop"),
    [string]$agentname
)

#Test path and create it if required

If(!(test-path $path))
{
      Write-Information `
            -MessageData "Path does not exist.  Creating Path..." `
            -InformationAction Continue;
      New-Item `
            -ItemType Directory `
            -Force `
            -Path $path | Out-Null;
      Write-Information `
            -MessageData "...Complete" `
            -InformationAction Continue
}
```

```
Set-Location $path

[Net.ServicePointManager]::SecurityProtocol = `
      [Net.SecurityProtocolType]::Tls12

Invoke-Webrequest `
      -uri https://packages.wazuh.com/3.x/windows/wazuh-agent-3.8.2-
1.msi `
      -outfile wazuh-agent-3.8.2-1.msi

Write-Host "Wazuh Downloaded..."

Invoke-WebRequest `
      -Uri https://raw.githubusercontent.com/aluminoobie/Install-
Wazuh/master/ossec.conf `
      -Outfile ossec.conf

Write-Host "Configuration File Retrieved..."

Write-Host "Installing Wazuh..."

.\wazuh-agent-3.8.2-1.msi /q `
      ADDRESS="172.20.3.35" `
      AUTHD_SERVER="172.20.3.35" `
      PASSWORD="ossec" `
      AGENT_NAME="$agentname" `
      /l*v installer.log

Copy-Item ossec.conf `
      -Destination 'C:\Program Files (x86)\ossec-agent\'

Restart-Service wazuh

Write-Host "Wazuh Installed!"
```

Appendix D: Install-Winlogbeat.ps1

```
<#
.SYNOPSIS
Install-Winlogbeat downloads Winlogbeat and installs Winlogbeat
with a configuration file.
.DESCRIPTION
PowerShell script or module to install Winlogbeat with configuration
.PARAMETER path
The path to the working directory.  The default is user Documents.
.EXAMPLE
Install-Winlogeat.ps1 -path C:\Users\example\Desktop
#>

[CmdletBinding()]

#Establish parameters for path
param (
    [string]$path=[Environment]::GetFolderPath("Desktop")
)

#Test path and create it if required

If(!(test-path $path))
{
      Write-Information `
            -MessageData "Path does not exist.  Creating Path..." `
            -InformationAction Continue;
      New-Item `
            -ItemType Directory `
            -Force `
            -Path $path | Out-Null;
      Write-Information `
            -MessageData "...Complete" `
            -InformationAction Continue
}

Set-Location $path

[Net.ServicePointManager]::SecurityProtocol = `
      [Net.SecurityProtocolType]::Tls12

Write-Host "Location set $path"

Write-Host "Retrieving Winlogbeat..."

Invoke-Webrequest `
      -Uri
https://artifacts.elastic.co/downloads/beats/winlogbeat/winlogbeat-
6.6.0-windows-x86_64.zip `
      -Outfile winlogbeat-6.6.0-windows-x86_64.zip

Write-Host "Winlogbeat Retrieved."

Write-Host "Retrieving Winlogbeat Configuration File..."
```

```
Invoke-WebRequest `
	-Uri https://raw.githubusercontent.com/aluminoobie/Install-
Winlogbeat/master/winlogbeat.yml `
	-OutFile winlogbeat.yml

Write-Host "Configuration File Retrieved."

Write-Host "Unzip Winlogbeat..."

Expand-Archive .\winlogbeat-6.6.0-windows-x86_64.zip `
	-DestinationPath 'C:\Program Files\'

Rename-Item `
	-Path 'C:\Program Files\winlogbeat-6.6.0-windows-x86_64' `
	-NewName 'C:\Program Files\Winlogbeat' `
	-force

Copy-Item winlogbeat.yml `
	-Destination 'C:\Program Files\Winlogbeat\' `
	-force

Write-Host "Unzip Complete.  Configuration File moved to C:\Program
Files\Winlogbeat\"

Set-Location -Path 'C:\Program Files\Winlogbeat\'

Write-Host "Installing Winlogbeat..."

.\install-service-winlogbeat.ps1

Start-Service winlogbeat

Write-Host "Winlogbeat Installed!"
```