Evaluating Open-Source HIDS with Persistence Tactic of MITRE Att&ck

Jon Chandler

Evaluating Open-Source HIDS with Persistence Tactic of MITRE ATT&CK

GIAC (GSEC) Gold Certification

Author: Jon Chandler, jchandler003@cox.net
Advisor: Sally Vandeven

Accepted: March 4, 2020

Abstract

Small companies with limited budgets need to understand if open-source tools can provide adequate security coverage. The MITRE ATT&CK framework provides an excellent source to evaluate endpoint security tool effectiveness. A MITRE research paper provides the following insight into the value of ATT&CK, "The techniques in the ATT&CK model describe the actions adversaries take to achieve their tactical objectives" (Strom, et al., 2019). This paper examines two open-source endpoint tools, OSSEC and WAZUH, against the MITRE ATT&CK framework. This analysis will determine each endpoint tool's ability to detect a select number of the MITRE ATT&CK framework persistence techniques. Out of the techniques reviewed, this paper will analyze the degree to which the ATT&CK technique can be accurately identified by the evaluated tools. MITRE also conducts evaluations but on proprietary tools. The results of the open-source endpoint tools analyzed here can be compared to the MITRE ATT&CK Evaluations conducted on the proprietary endpoint toolsets. The MITRE ATT&CK framework is a valuable methodology that allows a company to compare endpoint tools from a security risk and product evaluation perspective.

1. Introduction

A few years ago, during an annual penetration test for the researcher's employer, the penetration test team performed a purple team analysis utilizing the MITRE ATT&CK framework. This evaluation was not conducted to understand the threats to the corporate environment but to understand how the deployed cybersecurity tools and incident response team detected known attacks. The MITRE ATT&CK framework is a knowledge base of common tactics and techniques that attackers utilize to compromise a system. MITRE builds on real-world compromises. Each technique within the framework lists the known threat actors utilizing its vector. It is critical for any company to understand the importance of the return on investment of the cybersecurity tools and the information it provides to their cybersecurity teams to make informed decisions on security. As MITRE ATT&CK has grown in popularity, more companies are jumping on the bandwagon and touting how well their tool stands against the framework. For instance, "FireEye Endpoint Security delivered the highest efficacy scores, highest number of behavior-based detections, and provided the most relevant context in the 2018 MITRE ATT&CK™ assessment, announced on Feb. 13, 2019" (FireEye, 2019). Of course, this research will not attempt to determine the validity of the previously mentioned marketing material that summarizes the FireEye MITRE ATT&CK evaluation. The actual MITRE ATT&CK Evaluations can be found at the https://attackevals.MITRE.org website. All the best-known proprietary EDR solutions are either in the process or have had the ATT&CK Evaluation completed for a specific Advanced Persistent Threats (APT). These evaluations can then be used as a comparison point to determine which tools might be most advantageous to a given company. Of course, these evaluations cannot protect against unknown zero-day attacks. Based on this understanding, the MITRE ATT&CK framework is an excellent baseline that can be used to understand how the various cybersecurity tools alert and protect a company.

There is a push for companies to implement the MITRE ATT&CK framework. Per CSO Online, "ATT&CK can serve as a unifying taxonomy for different groups within an organization to share information, work together, and build the necessary detection and response procedures" (King, 2019). Companies are shifting security focus

to detection and response. Once a company understands what needs to be secured, they can utilize the framework to secure those areas. The MITRE ATT&CK framework helps companies know themselves and their detection abilities. The framework also helps the company understand the detected real-world compromise tactics and techniques and the enemies that utilize these steps.

For companies with large security budgets, it is typical to have high-priced endpoint detection and response (EDR) solutions for exposing malicious activity. However, for many smaller companies or companies with tight security budgets, these toolsets are prohibitively expensive. There are a few open-source projects that will cover some, if not all, of the functionality of the proprietary tools. For these smaller or budget-tight companies, it would be valuable to determine how effective these open-source toolsets are at identifying and monitoring system events that occur during a host compromise.

2. Research Tools

As mentioned previously, MITRE ATT&CK is a knowledge base of adversary tactics, techniques, and procedures (TTPs) based on real-world compromises. The ATT&CK site further explains the extent of the framework as, “The attack knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and the cybersecurity product and service community” (ATT&CK Matrix, n.d.). The framework constantly grows as research is regularly added. In order to further understand a technique, a research article describes it as, “The techniques in the ATT&CK model describe the actions adversaries take to achieve their tactical objectives” (Strom, et al., 2019). The framework as a baseline allows a comparison point between endpoint tools. One of these comparison points, MITRE ATT&CK evaluations have been conducted on the proprietary toolsets. However, evaluations of open-source tools are lacking.

At the time of this writing, MITRE has twelve different tactics defined. Each tactic includes from ten to seventy-three different techniques. It has a total of three

hundred and forty-three techniques. Each technique could have many different associated procedures.

This research identified two viable open-source Host Intrusion Detection Systems (HIDS): OSSEC and Wazuh. Wazuh was forked from the OSSEC project so there are many similar functions and configurations. Per the OSSEC website, users download their tool 500,000 times per year. The site describes, "OSSEC has a powerful correlation and analysis engine, integrating log analysis, file integrity monitoring, Windows Registry monitoring, centralized policy enforcement, rootkit detection, real-time alerting and active response. OSSEC runs on most operating systems, including Linux, OpenBSD, FreeBSD, MacOS, Solaris, and Windows" (OSSEC, n.d.). OSSEC is an open-source software, which means it is distributed for free based on the GNU General Public License.

The other open-source tool tested in this research is the Wazuh Host Intrusion Detection System (HIDS). Per the Wazuh website, "The Wazuh server is in charge of analyzing the data received from the agents, processing events through decoders and rules, and using threat intelligence to look for well-known IOCs (Indicators of Compromise)" (Wazuh, n.d.). Wazuh can analyze data from thousands of different agents. It can be set up in cluster mode which allows scalability. The server manages agents and can be used to remotely upgrade agents if desired. Wazuh can send remote commands to the endpoints. Wazuh is also an open-source tool that is freely distributed and "it provides new detection and compliance capabilities, extending OSSEC core functionality" (OSSEC vs Wazuh, n.d.).

StackShare.io, a web resource for developers and engineers, described OSSEC as a Host-based Intrusion Detection System and Wazuh as an Open Source Host and Endpoint Security. Features offered by OSSEC include Open Source HIDS, Multiplatform HIDS and PCI Compliance. Features offered by Wazuh include Security Analytics, Intrusion Detection and Log Data Analysis (OSSEC vs Wazuh, n.d.). These description differences were apparent during research as Wazuh included more data analytic tools with Kibana, Elk, Squil, Squert, etc. Figure 1 below demonstrates Wazuh's functionality including graphs and data analytics.

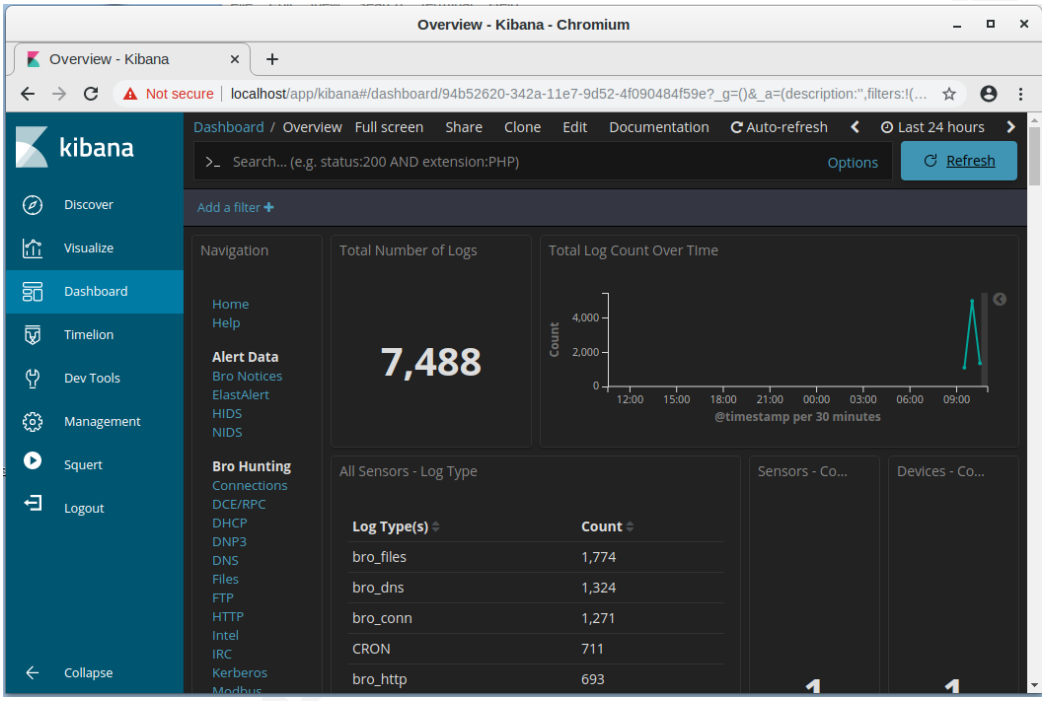

Figure 1 – Wazuh Monitoring Screen

On the other hand, OSSEC sports a rudimentary GUI that includes a Main, Search, Integrity checking, Stats and About informational tabs on the top of the page as seen in Figure 2 below. The basic search ability and stats accumulation offer handy tools but is far less advanced than the Wazuh counterpart.

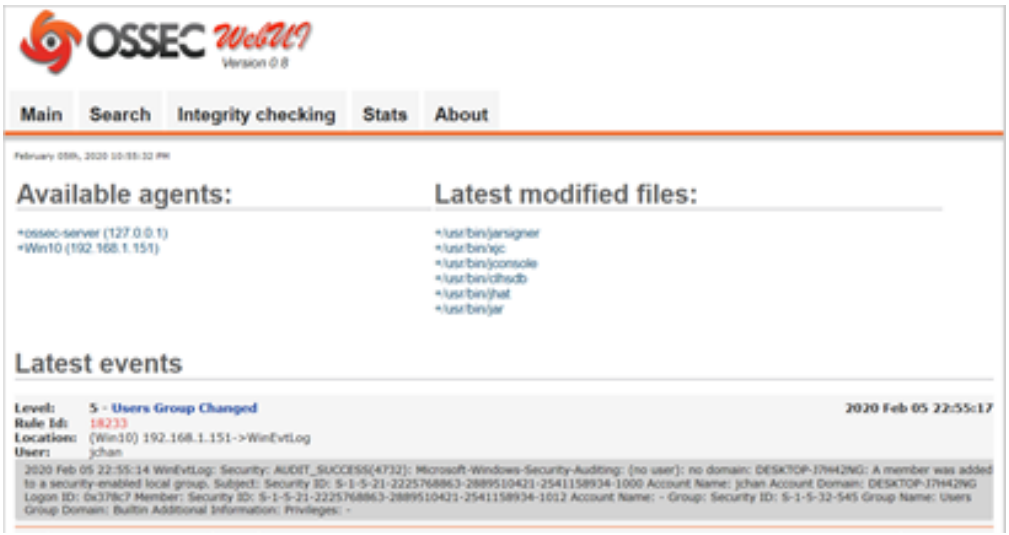

Figure 2 – OSSEC Monitoring Screen

Jon Chandler, jchandler003@cox.net

Other than the differences described above, it appears that the two engines have very similar backend functionality. However, the two Windows agent config files show many similarities and differences. These comparisons are for the default settings of each tool. Even though the defaults are different, many of the settings can be set in a similar fashion in both systems. For instance, the <client> section of both tool configuration files defines the server IP for the agent to communicate information. Each tool utilizes different names to define the server. OSSEC uses <server-ip> and Wazuh uses <address>. Wazuh has more configuration settings by default in the <client> section which includes settings for port, protocol, crypto_method, notify_time, time-reconnect, and auto_restart parameters. In all the OSSEC documentation there was no reference to the crypto_method to select between different encryption methods. The Wazuh documentation lists Blowfish and AES as encryption method options.

Other notable differences in the default setup include the initial files and registry key entries being monitored. The line entries were slightly different as Wazuh utilized the backslash while OSSEC utilized the forward slash as a file path delimiter. Despite the file path difference, the default-monitored files section of each of the config files include forty-five specific files that both tools monitor. Wazuh includes six lines of unique files to monitor while OSSEC had seventeen unique files. During testing, the researcher identified that files that are necessary to identify changes for the specific test conducted as part of this research were not always included by default. For instance, sethc.exe is one of the files that can be changed in the Accessibility Features test. The sethc.exe file was included by default in Wazuh but not in OSSEC. The registry settings were the same between the two config files for eighteen registry keys. Wazuh included twenty unique registry keys while OSSEC only included ten unique keys. Either tool could easily include all the possible registry keys in both files including the unique entries to each tool.

To further compare the two endpoint tools, the rootcheck section of the config file was very similar in both Wazuh and OSSEC, with only one unique line to each config section. The OSSEC tool included a line for the win_audit_rcl.txt file. This file is used to check registry values and is available in both systems but is only included in the OSSEC

by default. The Wazuh tool included a line called disabled which implies that the rootcheck can be enabled or disabled. Figure 3 and Figure 4 show the differences in the two rootcheck sections of each tool.

```
<!-- Rootcheck - Policy monitor config -->
<rootcheck>
  <windows_audit>./shared/win_audit_rcl.txt</windows_audit>
  <windows_apps>./shared/win_applications_rcl.txt</windows_apps>
  <windows_malware>./shared/win_malware_rcl.txt</windows_malware>
</rootcheck>
```

Figure 3 – OSSEC Rootcheck

```
<!-- Policy monitoring -->
<rootcheck>
  <disabled>no</disabled>
  <windows_apps>./shared/win_applications_rcl.txt</windows_apps>
  <windows_malware>./shared/win_malware_rcl.txt</windows_malware>
</rootcheck>
```

Figure 4 – Wazuh Rootcheck

As mentioned in this section, OSSEC and Wazuh have many similarities and differences. The fact that Wazuh originated from OSSEC helps us understand the reason for the similarities. Many of the differences stem from Wazuh's attempts to provide extra features such as the improved GUI interface and the ability to remotely update agents. Next, this paper will discuss of how the research will utilize the MITRE ATT&CK persistent tactics to understand the effectiveness of the OSSEC and Wazuh tools.

3. MITRE ATT&CK

3.1. Persistence Tactics

Attackers are continually finding new ways to persist on compromised systems in the hopes of avoiding detection. The MITRE ATT&CK framework constantly updates the framework based on new research and detected compromises. ATT&CK was selected for this research because of its all-inclusive model created from real-world compromises. It also serves as a baseline during the testing process. For this research, it's not feasible to test all three hundred and forty-three techniques. Therefore, due to the plethora of TTPs, a select sample of thirty techniques under the persistence tactic will be utilized to test how well OSSEC and WAZUH can identify the techniques and procedures of the

persistence tactic. The attacker must maintain persistence by altering the victim system in a detectable way. ATT&CK outlines how attackers alter the victim's system.

The MITRE ATT&CK framework is outlined on the attack.mitre.org website. It must be pointed out that all these mechanisms require administrative access to the Windows 10 system. Understanding how the attackers gain administrative access is beyond the scope of this paper. The following is a list of the techniques selected from the persistence tactic. For definitions of each of these techniques, refer to Appendix II.

1. Accessibility Features
2. AppCert DLLs
3. Appinit DLLs
4. Application Shimming
5. Auhentication Package
6. BITS Jobs
7. Bootkits
8. Change Default File Association
9. Create Accounts
10. DLL Search Order Hijacking
11. Hidden Files and Directories
12. Hooking
13. Image File Execution Option Injection
14. Logon Scripts
15. LSASS Driver
16. Modify Existing Service
17. Netsh Helper DLL
18. New Service
19. Path Interception
20. Port Monitoring
21. PowerShell Profile
22. Redundant Access
23. Registry Run Keys / Startup Folder
24. Schedule Tasks
25. Screensaver
26. Security Support Provider
27. Service Registry Permission Weakness
28. Shortcut Modification
29. Windows Management Instrumentation Event Subscription
30. Winlogon Helper DLL

The researcher analyzed each of the selected techniques in the MITRE ATT&CK framework to categorize the underlying Windows 10 system changes utilized in the technique. Through this analysis, the researcher identified the following change categories/methods required to maintain Windows persistence from the selected thirty techniques:

- Registry addition or change

- File addition or change
- Account usage, creation, or manipulation
- Application shimming
- Master boot record alteration

By categorizing changes as outlined above, this paper will eliminate redundancy as most of the techniques could utilize registry additions and changes. The reverse is also true in that a technique may fall under multiple change categories as there may be multiple ways in which to accomplish the technique's objective. For instance, Accessibility Features can either change a file or a registry key to accomplish its objectives.

3.2. Documentation Methodology

The change categories identified above will be used to document the testing for this paper instead of documenting techniques individually. For instance, many of the techniques alter registry entries and the HID tools should be able to identify these changes at the registry level despite the technique utilized. Furthermore, techniques may utilize more than one of the change categories, therefore it would be redundant to list a technique and how the tool identifies a registry change and them move to the next technique and list the same conclusion about registry changes through all the techniques that alter a registry key. Out of the thirty techniques tested, a majority can be exploited with either registry changes or file changes. Methods will be utilized to change one change category at a time. After the test, each HIDS tools will be analyzed to determine if the categorical change was identified.

The bootkit technique adds another dimension to the change categories. This single technique typically requires multiple categories at the same time. For instance, a bootkit may require registry changes, file changes, and master boot record alterations altogether in order to install. The previous paragraph explains the changes as either of the two categories to accomplish the technique objective while bootkits alters more than one category at the same time. This is a generalization as each bootkit has its own method of compromising a system. However, for the

purposes of this research, bootkits will be viewed holistically instead of broken out into the subsets of changes. The master boot record changes will be considered only as part of a bootkits. Bootkits fall under a unique process and will be documented individually as multiple categories of changes are coupled together in the malware. The malware that was selected to test this technique includes registry, file and master boot record changes.

The testing conducted against the thirty techniques utilized regular built-in Windows functionality such as command prompt or PowerShell. This is like "living off the land" for real attackers. Tools that are already available on a Windows machine are much less likely to raise a red flag when executed. Albeit, Windows functionality such as command prompt and PowerShell are typically only executed by administrators.

Several organizations have created test scripts or automated tools to complete testing against the MITRE ATT&CK framework. One such option is the Purple Team ATT&CK Automation tool by Praetorian. This method requires a payload.exe file created from msfvenom to be run on the Windows 10 test machine. Praetorian recommended whitelisting the binary in order to complete testing. At the time of writing, the Praetorian Red Team Automation tool automated one hundred and nine techniques across all the tactics. Another option more suitable for this research is a set of tools created by the Atomic Red Team as hosted on GitHub. The "Atomic Red Team allows every security team to test their controls by executing simple 'atomic tests' that exercise the same techniques used by adversaries" (Red Canary Co., n.d.). These Atomic Red Team tests utilize the underlying change categories mentioned above to make the Windows system configuration changes. Therefore, the researcher utilized the Atomic Red Team scripts for the selected testing.

4. Findings and Discussion

In order to provide more context around the Atomic Red Team testing scripts, an explanation must be presented about the specific scripts utilized. In many of the scripts, the Reg.exe command changed registry entries from the normal functionality to

something abnormal. For instance, one item utilized the following command to cause an executable to run at startup for the Registry Run Keys / Startup Folder test.

```
REG ADD "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /V "Atomic Red
Team" /t REG_SZ /F /D "c:\temp\T1060-notepad.exe"
```

Notice that this change did not introduce malicious content but utilized the Reg command to cause the Notepad.exe to run at startup. The same concept could be utilized to cause the command prompt to execute instead of the normal Accessibility Features. Changing the Accessibility Features to launch the command prompt allows an attacker to gain access to the system prior to authentication. As can be seen, the test demonstrates that an attacker could utilize a Windows command to change functionality to run their intended objectives.

The changed functionality does not need to be defined as malicious (IOC) as it could include any functionality benefiting their intentions which could include a zero-day exploit. The Notepad program mentioned above could easily be replaced by background-running malicious code that runs unnoticed by the user. Therefore, the true objective of the test is to understand if a change can be made to a specific configuration item within Windows 10 and how the HIDS tools identify or alert on those changes. The results of the tests will be posted in Appendix I as a supplement. The analysis will focus on the categories of changes allowed to the system such as registry or file changes. Focusing on category of changes helps with the redundancy of testing the techniques specifically. For instance, registry key change tests repeat multiple times for each of the techniques that utilize that category of changes. The Accessibility Features, AppCert DLLS, Appinit DLLs, and many the other techniques alter registry keys to accomplish malicious intent.

The paper will describe how each of the categories were used to accomplish the technique objectives. Please note that some techniques could be accomplished in different ways. For instance, Accessibility Feature can be changed by either the registry or by copying files. The analysis will then explain how the tools identified the specific category of change. It will also identify the shortcomings and strengths associated with each area. It is important to note that alert rules were not required due to the techniques that were

chosen. If a technique was not detected by the HIDs, research was conducted to determine what needed to be added to the configuration files.

4.1. Registry Additions or Changes

The first change category that will be analyzed is the registry key updates. The majority of the techniques (seventeen out of thirty) can utilize registry edits to accomplish the objective of the specific technique procedure. Both OSSEC and Wazuh include configuration file entries that defines which registry keys to monitor, although it was necessary to add specific registry key entries during tested. Both HIDS offer a similar setup regarding registry monitoring. The following line associated with a service registry changes is listed under the SysCheck section of the configuration file.

```
<windows_registry>HKEY_LOCAL_MACHINE\System\CurrentControlSet\
Services</window_registry>
```

Per both OSSEC and Wazuh's documentation for the latest instances, the windows_registy tag states, "new entries will not trigger alerts, only changes to existing entries" (OSSEC, n.d.). For these changed registry keys, the HIDS capture both MD5 hash and SHA1 hash to compare against historic values. When a registry change is detected, the HIDS generates a Syscheck alert in the log files. The specific rule is 'Registry Integrity Checksum Changed' for that specific registry key. However, the alert doesn't provide specifics about what changed. Figure 5 below shows an example of an OSSEC Registry Integrity Checksum Changed alert. The Wazuh alert shows the same information just in a different format.

Figure 5 – Trying to find registry key for CurrentVersion/Run

You can see in this alert that the registry key that changed was listed. However, the alert doesn't show exactly what changed within the key.

An example of a registry key change is the Winlogon Helper DLL test. In this test, neither the old nor the new command associated with that registry key was provided. The only details in the logs showed the Integrity checksum changed for the following registry key.

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\
USBHUB3\Enum
```

In the Accessibility Features technique, neither the osk.exe, utilman.exe. magnify.exe, narrator.exe nor the sethc.exe files were identified as being changed in the registry key. The cmd.exe or calc.exe files that replaced the legitimate Accessibility Features files were not identified either.

The vagueness of the alert leads to the conclusion that it would be very difficult for an incident responder to determine what changed in the system. Although the specific registry keys reviewed should not fluctuate over time, they typically remain static after installation, except when Windows is updating. The potential for false positives in this category is small as the tool does not alert on new registry entries—it will only alert on registry changes. However, the researcher noted that when tools such as Windows Defender are running, more noise occurs in the registry changes. Windows Defender automatically reverted some of the testing changes back to original after they executed.

Please note that a monitored registry key can stop short of the final subkey. This means that multiple subkeys could exist under a key that is being monitored. For instance, on the Windows 10 test VMs, the following OSSEC configuration file entry includes six different subkeys.

```
<windows_registry>KEY_LOCAL_MACHINE\Software\Microsoft\
Windows\CurrentVersion\Policies</windows_registry>
```

If a ‘Registry Integrity Checksum Changed’ alert is generated, it will take time to understand which are the unchanged registry keys and which are the malicious changes. In the Windows 10 test VM, the following only has one value under the key.

```
<windows_registry>HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\
CurrentVersion\Run</windows_registry>
```

Although a registry key with only limited subkeys would be less tedious, it would still take time to understand what is normal and abnormal. A savvy attacker would take every effort to ensure that additional changes to the registry keys appear as normal as they can in order to avoid detection. This could include only making a one-character change that is hard to detect such as changing to a character that looks like the original character. Therefore, the Incident Response analyst following up on the alert might not recognize the insignificant alteration to the changed registry key. The problem could be exasperated depending on how many false positives are identified by the system. Only select registry keys are entered into the configuration file to ensure the number of false positives are kept minimal.

4.2. File Additions or Changes

File additions and changes are difficult to monitor. OSSEC and Wazuh both allow monitoring on either directory or a specific file basis. On a directory monitoring basis, a company can set the configuration to monitor for directory changes but unless the folder only contains relatively few unchangeable files, the number of false positives from a security perspective could be very large. Applications don't always separate executable files from data files within a folder or directory. Even if they are separated, the potential to create temporary files while the application is running could cause unwanted false positives in monitoring. It would create too much noise to monitor everywhere on a drive or volume and the HIDS tool would generate hundreds of alerts per endpoint just for regular system usage. As a test to determine the number of false positives that would be created, the researcher set monitoring for just the Windows folders and noticed several alerts that were generated due to temporary file creation and regular system usage. As for specific file monitoring, it is cumbersome to create entries for hundreds of specific files on the system and monitoring specific files will not capture new files added to a directory. Therefore, one can see that monitoring file additions and changes requires a fine line between monitoring specific files and specific directories.

The potential to add or change a file occurs in nearly all the selected sample of ATT&CK techniques. There are many tools normally available to a user to perform administrative tasks on a Windows system. Utilizing these available tools can add or

change typical functionality to incur malicious intent on a system. A normal Windows 10 tool such as BITS Jobs exist to perform remote downloads from the internet, “In fact, malware-free attacks in general have surged in recent years, accounting for 40 percent of the total number of cyberattacks globally last year: according to the 2019 CrowdStrike® Global Threat Report, attackers continue to shift to defense evasion methods, like living off the land techniques, to remain undetected” (Goudie, 2019). However, at some point in the attack process a file change will be required on a Windows system to accomplish the full intent of the comprise. The only problem is that this is the weakest detection potential due to the plethora of already changing files on a system and the many locations in which files can be placed and hidden. If detection teams can focus on the high-risk files which consists of a small number of folders such as c:\windows\system32 or a small number of specific high-risk files such as the Accessibility Feature files to monitor, they would detect a larger percentage of attacks. They would have to rely on other mechanisms to detect attacks that don’t utilize the high-risk files. Teams must realize that they are only detecting a small percentage of the potential changed files, especially since many of the registry entries can easily change file locations to anywhere on the disk. Focusing on the highest risk files and folders would be the appropriate actions to take with the understanding that this method/technique is not foolproof.

Both OSSEC and Wazuh utilize similar line structure to monitor specific files. As mentioned previously, OSSEC separates items in the path by a forward slash while Wazuh utilizes the backslash. The out-of-the-box configuration of both OSSEC and Wazuh have a specific set of sensitive Windows files being monitored. Each system also includes a subset of unique files that HIDS deems important. If desired, a file or folder could be easily added to each of the monitoring configurations. During testing, it was required to add entries into the configuration file of each HIDS to cover the risks of the MITRE ATT&CK framework. Again, like the registry monitoring, this functionality does an MD5 and SHA1 to compare current files with historic files. The system regularly checks the file integrity to determine if changes were made. Once a change is detected, an alert is generated in the alerts log file (and email alerts are generated, if configured to do so).

The researcher performed two tests for this section. The first test copied the cmd.exe file over the sethc.exe file. The change required booting up to a bootable CD that could still access the local hard drive of the image. Once the temporary system was activated, a command prompt could be launched to copy the files as mentioned above. Once the system was rebooted and waiting to login, the operator could press the shift key five times to cause the StickyKeys window to appear. However, since the StickyKey program was replace by the cmd.exe, the command prompt executed when the shift key combination executed. This allows a user to gain access to the system bypassing authentication as the command prompt executes in system mode. A hacker could then copy files to the system or change the password for the administrator account. The following line caused OSSEC to monitor this file change:

```
<directories check_all="yes">%WINDIR%/SysNative/sethc.exe</directories>
```

After logging back into the system, the alert in Figure 6 showed up in the alerts log file.

```
** Alert 1581380401.170137: mail  - ossec,syscheck,
2020 Feb 10 16:20:01 (Win10) 192.168.1.151->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: 'C:\WINDOWS/SysNative/sethc.exe'
Size changed from '100352' to '280064'
Old md5sum was: '0899a093b921e175dcf7ea5ebcc65306'
New md5sum is : 'd7ab69fad18d4a643d84a271dfc0dbdf'
Old sha1sum was: 'b4ffbf9b6404fb089c9c9781f773ecd02fd0b7bc'
New sha1sum is : '8dca9749cd48d286950e7a9fa1088c937cbccad4'
```

Figure 6 – Log Alert for sethc.exe File Change

It is noted that the alert shows the file size changes as well as the checksums from the old and new files. Unlike the registry changes that require additional research to determine what changed, this alert is very clear and shows the exact altered file.

The second test copied a file to the Startup folder. The system only allows administrative access to copy files to the Startup folder. The researcher copied the calc.exe file to the folder listed below:

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
```

The configuration file for OSSEC contained the following entry to monitor the additions to the Startup folder.

```
< directories check_all="yes" realtime="yes">%PROGRAMDATA%/
Microsoft/Windows/Start Menu/Programs/Startup</directories>
```

When the machine rebooted, the calculator automatically executed. Obviously, this change could have included malicious code via a batch script or another executable program type. Regardless of the executable method, the program runs if an attacker can add to this folder. Figure 7 shows the alert that generated in the alerts.log file. Since this is a new file, comparisons cannot be completed from hashes. However, this alert demonstrates simply the addition of a file to the Startup folder.

```
** Alert 1581379144.124147: mail  - ossec,syscheck,
2020 Feb 10 15:59:04 (Win10) 192.168.1.151->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: 'C:\ProgramData/Microsoft/Windows/Start Menu/Programs/Startup/calc.exe'
Size changed from 'CreateFile=32' to '27648:33279:S-1-5-32-
544:0:f88cc05134c555d4e1cd1def78162a9a:c882f09eafddb75843bddbacb796b90195445183'
```

Figure 7 – Results of the calc.exe file added to the Startup folder

4.3. Account Additions

Account addition adds a new account to the Windows 10 environment. If unnoticed, the attacker has continual system access. Windows 10 logs the creation of new accounts. The Windows logs are transferred to both OSSEC and Wazuh. Both HIDS integrate monitoring the Windows logs. They also both detect password changes. However, this category consisted only of the Account Creation technique. The difference in the two HIDS configuration files can be seen in Figure 8 and Figure 9 below. The OSSEC file references eventlog while Wazuh references eventchannel. The Wazuh documentation lists that eventlog supports every version of Windows while eventchannel supports Windows Vista and later (Wazuh, n.d.). Also, Wazuh configuration file is specifically excluding a list of specific events from its log file.

```
<!-- One entry for each file/Event log to monitor. -->
<localfile>
  <location>Application</location>
  <log_format>eventlog</log_format>
</localfile>

<localfile>
  <location>Security</location>
  <log_format>eventlog</log_format>
</localfile>

<localfile>
  <location>System</location>
  <log_format>eventlog</log_format>
</localfile>

<localfile>
  <location>Windows PowerShell</location>
  <log_format>eventlog</log_format>
</localfile>
```

Figure 8 – OSSEC Event Log Monitoring

```
<!-- Log analysis -->
<localfile>
  <location>Application</location>
  <log_format>eventchannel</log_format>
</localfile>

<localfile>
  <location>Security</location>
  <log_format>eventchannel</log_format>
  <query>Event/System[EventID != 5145 and EventID != 5156 and EventID != 5447 and
    EventID != 4656 and EventID != 4658 and EventID != 4663 and EventID != 4660 and
    EventID != 4670 and EventID != 4690 and EventID != 4703 and EventID != 4907 and
    EventID != 5152 and EventID != 5157]</query>
</localfile>

<localfile>
  <location>System</location>
  <log_format>eventchannel</log_format>
</localfile>

<localfile>
  <location>active-response\active-responses.log</location>
  <log_format>syslog</log_format>
</localfile>
```

Figure 9 – Wazuh Event Log Monitoring

In both HIDS, they easily identified account creations. The Atomic Red Team test plan for Account Modification was a regular Windows 10 command to add a user from the

command prompt. The following line was executed as administrator on each of the Windows 10 test machine.

```
net user /add eviluser
```

After the command was run, the tester ran the net user command to confirm the added user. After verification, a review of the logs confirmed that the system identified the added account. Figure 10 below shows four entries created in the alert log.

Figure 10 – Results of an account creation

Two sets of logs identify a user account created and user account changed. The second alert in each set specifically names eviluser as the created account. The amount of information in this alert makes it difficult to identify the created account as the account name is nested in the middle of the text.

4.4. Application Shimming

Application Shimming test plans include three different tests. The first test consists of running the sdbinst.exe file to import a database into the Application Shimming database as follows:

```
sdbinst.exe AtomicShimx86.sdb
```

As mentioned above, Application Shimming is for backwards compatibility. Companies may want to copy an Application Shimming database across their environment to ensure all machines can correctly run older software. Obviously, the sdbinst.exe file is adding configuration information for legacy applications to be able to continue to run. The sdbinst.exe is just copying files to the Application Shimming system location. The other two methods specifically set a registry entry or copy a file to the system location as shown below.

```
Method 1
New-ItemProperty -Path HKLM:"\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\AppCompatFlags\Custom" -Name "AtomicRedTeamT1138" -Value
"AtomicRedTeamT1138"
New-ItemProperty -Path HKLM:"\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\AppCompatFlags\InstalledSDB" -Name "AtomicRedTeamT1138" -
Value "AtomicRedTeamT1138"
```

```
Method 2
Copy-Item $PathToAtomicsFolder\T1138\bin\T1138CompatDatabase.sdb
C:\Windows\apppatch\Custom\T1138CompatDatabase.sdb
Copy-Item $PathToAtomicsFolder\T1138\bin\T1138CompatDatabase.sdb
C:\Windows\apppatch\Custom\Custom64\T1138CompatDatabase.sdb
```

The above sections (Registry Changes and File Changes) covered the Application Shimming category.

4.5. Bootkits / Rootkit

According to Kasperky's Encyclopedia, a rootkit is "A program or collection of software tools programmed to hide certain objects or activity in the system. As a rule, cybercriminals hide the registry keys for the autorun of malicious objects, as well as files, folders, processes in the infected computer's memory, and malicious network activity" (Rootkit, n.d.). The same website described "a bootkit is a malicious program designed to load as early as possible in the boot process, in order to control all stages of the operating system start up, modifying system code and drivers before anti-virus and other security components are loaded" (Bootkit, n.d.). Bootkits can load malicious code in the Master Boot Record (MBR) or boot sector. Both OSSEC and Wazuh have been designed to detect Rootkits. However, nothing is noted in their documentation about Bootkits. Therefore, this test will focus on running a rootkit on the test machines to determine if OSSEC and Wazuh detect its existence.

The tester utilized the Rootkit.Win32.TDSS.tdl4 (TDSS) rootkit provided in the SANS course SEC501 in a package called badkit3.exe. SEC501 describes the badkit3.exe as a Master Boot Record (MBR) Rootkit. When installed in administrative mode, the rootkit caused the system to blue screen with a message that stated, "Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you." The system rebooted after the blue screen message. Both the Wazuh and OSSEC Windows 10 machines created the same blue screen error messages. Both Wazuh and OSSEC reported anomalies due to the rootkit. Figure 11 below shows that Wazuh identified alerts related to the rootkit.

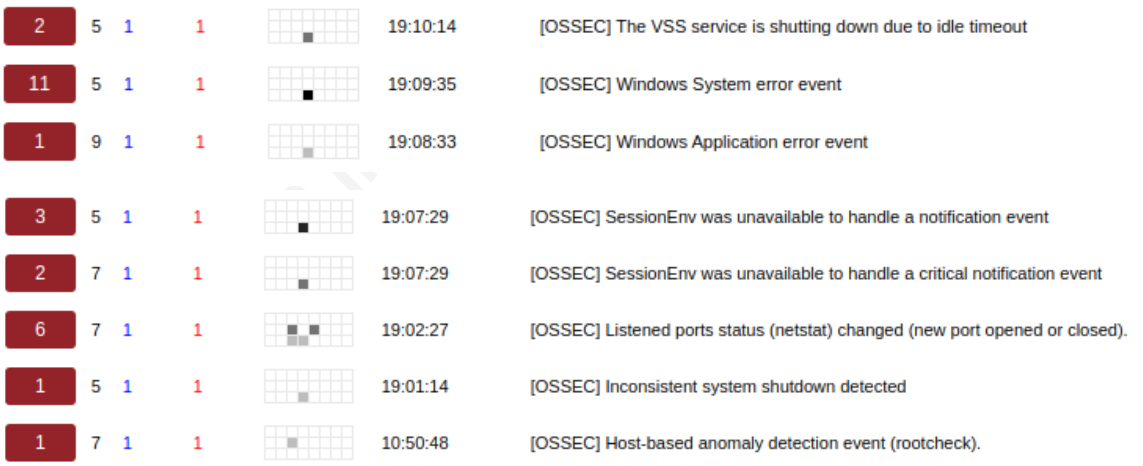

Figure 11 – Wazuh Alerts for Rootkit

OSSEC showed similar error messages that the malware caused system problems. See Figure 12 below of the OSSEC error alert generated. This alert only shows that Windows error was generated due to the rootkit changing files.

Figure 12 – OSSEC alert generated due to malware cause a system error

However, neither tool directly identified the malware. The TDSKiller program provided in SEC501 effectively identified and removed the Rootkit.Win32.TDSS.tdl4 malware as Figure 12 below shows.

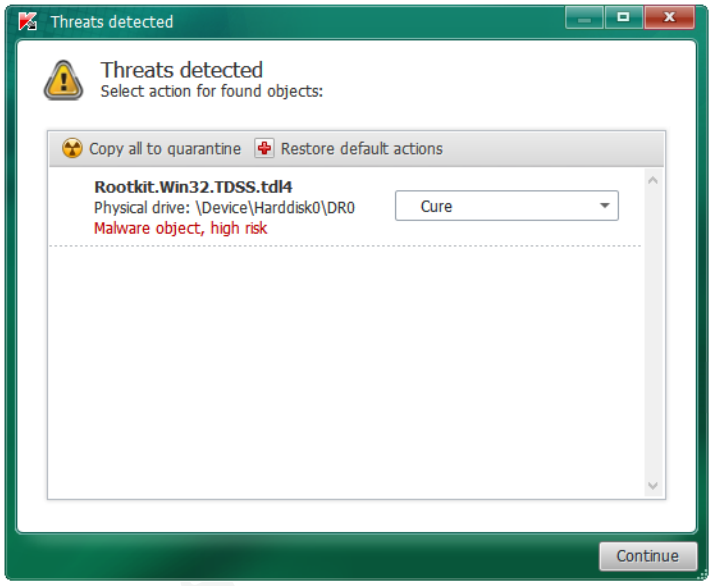

Figure 12 – TDSSKiller Removing MBR Rootkit

The HIDS did not effectively detect this MBR Rootkit. It showed signs that a problem existed and that something suspicious happened on the infected PC. However, the HIDS did not provide a clear description of the malware. In fact, the Incident Response (IR) team would struggle to understand the true alerts associate with an MBR Rootkit. In this case, suspicion would most likely generate enough concern that the IR team's utilization of other tools would identify the root cause of the issue.

5. Proprietary Tool Comparison

A comparison of OSSEC and Wazuh results from above can be compared to the MITRE ATT&CK Evaluation as found at the following link.

https://attackevals.mitre.org/evaluations/crowdstrike.1.apt3.1.html

On this site, a section evaluates Crowdstrike against the entire framework (MITRE ATT&CK, n.d.). In the persistence column, Crowdstrike detected the following techniques: Accessibility Features, Create Account, New Service, Registry Run Keys /

Startup Folder, Scheduled Task, and Valid Account. For instance, the evaluation provided the information in Figure 13 for the Accessibility Features of the Crowdstrike Evaluation.

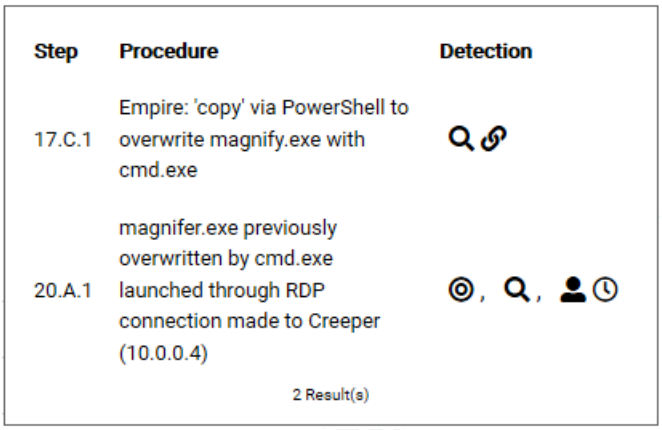

Figure 13 – Crowdstrike's ATT&CK Evaluations – Accessibility Features

This figure shows a procedure followed by different detection techniques. For 17.c.1, the magnifying glass stands for telemetry and the other symbol stands for tainted. For 20.A.1, the symbols represent specific behavior, telemetry, and general behavior (delayed). If the user clicks on the symbols, more information is provided. The screen shows that "Telemetry showed a file write of magnify.exe by powershell.exe in the system directory. The telemetry was tainted by an alert on its parent powerhsell.exe process" (MITRE, n.d.). Because the research conducted here and the evaluation conducted by MITRE did not follow the exact same process, it is uncertain if similar comparisons were made. Out of the methods utilized by MITRE, we do not have a full understanding of what the stipulations were for the tests. Based on the website, the MITRE evaluation process included legitimate compromises instead of tests to see if defined functions and commands already available on a Windows 10 machine could be altered and if they continue to work. In Appendix I, the table shows that OSSEC and Wazuh have four detections and two partial detections. This table also shows that Crowdstrike has two detections and three partials. Again, the result doesn't reflect the exact same testing methodology.

According to the testing as outlined in Appendix I, the HIDS tested here fared just as well, if not better (depending on testing methodology) with a proprietary tool like

Crowdstrike. However, there are pros and cons to proprietary tools versus open-source tools. Proprietary tools typically offer configuration and support as part of the product purchase. The company has the expertise and can help resolve issues when they arise as part of the support model. This is built into the price for the software and support. The issues identified working with open-source tools are common complaints that result from using free tools. If you have a problem, you are on your own to resolve the issue. Employees that support open-source tools need to be experts on the systems. It was noticed that companies do exist that charge a fee to provide support for their version of OSSEC.

During the testing, many issues came up that caused problems for the researcher to understand and resolve. Granted, the researcher is new to these tools and had a steep learning curve understanding how to install and configure the tools. Installing supplemental functionality like adding the GUI for OSSEC or Kibana / Elk for Wazuh was problematic. The configuration of each of the tested tools requires different skill levels and experience for the employee supporting the tools. The small window of functionality that was tested here is miniscule compared to the full functionality required to protect an organization. During the testing, no additional rules were required as the focus was on persistence and the Atomic Red Team testing methodology didn't require actual compromises to complete the testing which would have necessitated additional rules for detection.

6. Conclusion

Based on the evaluation completed the researcher saw that both OSSEC and Wazuh function similarly. The two test system engines appear to function similarly. The configuration files offer much of the same monitoring capabilities. Both HIDs provided similar functionality from the detection perspective. They have the ability to detect rootkits, specific file changes, registry changes and account creation and modification.

The main difference between the two tools lies in their external functionality. Wazuh offered cluster mode that allows scalability; however, this functionality wasn't mentioned for OSSEC. As mentioned in the sections above, the Wazuh server manages

agents and can be used to remotely upgrade agents if desired. Wazuh can send remote commands to the endpoints. This functionality appears to be only available in Wazuh. The data analytic functionality appeared to be much more advanced for Wazuh compared to OSSEC. These are only a few differences that were noticed during testing, but it is certain that many other differences between the two HID systems exist.

As for comparison to proprietary tools, it appears from the surface that both tools function well in the security detection arena. According to the testing, with the right configurations, many additional registry changes were identified than what was identified for Crowdstrike. However, if an attack resulted in both file changes and registry changes, it would be apparent that the identification would quickly dwindle. Both tools struggled with file detection based on the location of the changed file. If the Accessibility Features file change requires it to be in a specific location such as c:\windows\system32, detection is certain.

For many smaller companies, open-source tools may be a great alternative to expensive EDR solutions. It would be critical for a small company thinking about OSSEC or Wazuh to fully evaluate the resources required to support an open-source tool. Internal human resources to support open-source tools can add up which means the open-source tools are not free of costs. Errors in monitoring and detection could be very costly for less experienced cybersecurity teams. Companies need to fully analyze their options and assess their risk appetite to understand the monitoring tools that best meet their company's objectives.

References

AppInit DLLs and Secure Boot - Win32 apps. (2018, May 30). Retrieved January 25, 2020, from https://docs.microsoft.com/en-us/windows/win32/dlls/secure-boot-and-appinit-dlls

ATT&CK Matrix for Enterprise. (n.d.). Retrieved January 25, 2020, from https://attack.MITRE.org/

Bejtlich, R. (2019, March 28). Thoughts on OSSEC Con 2019. Retrieved January 25, 2020, from https://taosecurity.blogspot.com/2019/03/thoughts-on-ossec-con-2019.html

Bootkit. (n.d.). Retrieved January 25, 2020, from https://encyclopedia.kaspersky.com/glossary/bootkit/

Cid, D. (2015, December 9). Server Security: OSSEC Updated With GeoIP Support. Retrieved January 25, 2020, from https://blog.sucuri.net/2015/12/ossec-with-geoip.html

FireEye Endpoint Security Leads MITRE ATT&CK Evaluation. (2019, February 19). Retrieved January 25, 2020, from https://www.fireeye.com/blog/products-and-services/2019/02/MITRE-evaluation-validates-fireeye-endpoint-security-as-most-effective-edr-solution.html

Goudie, M. (2019, May 9). The Rise of "Living off the Land" Attacks. Retrieved January 25, 2020, from https://www.crowdstrike.com/blog/going-beyond-malware-the-rise-of-living-off-the-land-attacks/

Kang, A. (2019, May 21). How to implement and use the MITRE ATT&CK framework | CSO ... Retrieved January 25, 2020, from

https://www.csoonline.com/article/3396139/how-to-implement-and-use-the-mitre-attandck-framework.html

MITRE ATT&CK. (n.d.). Retrieved January 25, 2020, from https://attackevals.mitre.org/evaluations/crowdstrike.1.apt3.1.html

OSSEC. (n.d.). Retrieved January 25, 2020, from https://www.ossec.net/

Ossec vs Wazuh: What are the differences? (n.d.). Retrieved February 5, 2020, from https://stackshare.io/stackups/ossec-vs-wazuh

Red Canary Co. (n.d.). Retrieved January 25, 2020, from https://github.com/redcanaryco/atomic-red-team

Rootkit. (n.d.). Retrieved January 25, 2020, from https://encyclopedia.kaspersky.com/glossary/rootkit/

Strom, B. (2019, July 17). Getting Started with ATT&CK: Adversary Emulation and Red Teaming. Retrieved October 21, 2019, from https://medium.com/mitre-attack/getting-started-with-attack-red-29f074ccf7e3

Strom, B. E., Battaglia, J. A., Kemmerer, M. S., Kupersanin, W., Miller, D. P., Wampler, C., … Wolf, R. D. (2019, October 11). Finding Cyber Threats with ATT&CK-Based Analytics. Retrieved October 19, 2019, from https://www.mitre.org/publications/technical-papers/finding-cyber-threats-with-attck-based-analytics.

Wazuh. (n.d.). Retrieved January 25, 2020, from https://wazuh.com/

Strunk, W., & White, E. B. (1999). *The elements of style*. Boston: Allyn and Bacon.

Appendix I

Rubric - Persistence Vector	OSSEC	WAZUH	Crowdstrike
1. Accessibility Features	Partial	Partial	Partial
2. AppCert DLLs	Not Detected	Not Detected	Not Detected
3. AppInit DLLs	Not Detected	Not Detected	Not Detected
4. Application Shimming	Not Detected	Not Detected	Not Detected
5. Authentication Package	Not Detected	Not Detected	Not Detected
6. BITS Jobs	Not Detected	Not Detected	Not Detected
7. Bootkit	Not Detected	Not Detected	Not Detected
8. Change Default File Association	Detected	Detected	Not Detected
9. Create Account	Detected	Detected	Detected
10. DLL Search Order Hijacking	Not Detected	Not Detected	Not Detected
11. Hidden Files and Directories	Not Detected	Not Detected	Not Detected
12. Hooking	Detected	Detected	Not Detected
13. Image File Execution Options Injection	Not Detected	Not Detected	Not Detected
14. Logon Scripts	Not Detected	Not Detected	Not Detected
15. LSASS Driver	Not Detected	Not Detected	Not Detected
16. Modify Existing Service	Partial	Partial	Not Detected
17. Netsh Helper DLL	Not Detected	Not Detected	Not Detected
18. New Service	Not Detected	Not Detected	Partial
19. Path Interception	Not Detected	Not Detected	Not Detected
20. Port Monitors	Not Detected	Not Detected	Not Detected
21. PowerShell Profile	Not Detected	Not Detected	Not Detected
22. Redundant Access	Not Detected	Not Detected	Not Detected
23. Registry Run Keys / Startup Folder	Not Detected	Not Detected	Detected
24. Scheduled Task	Not Detected	Not Detected	Partial
25. Screensaver	Not Detected	Not Detected	Not Detected
26. Security Support Provider	Not Detected	Not Detected	Not Detected
27. Service Registry Permissions Weakness	Not Detected	Not Detected	Not Detected
28. Shortcut Modification	Detected	Detected	Not Detected
29. Windows Management Instrumentation Event Subscription	Not Detected	Not Detected	Not Detected
30. Winlogon Helper DLL	Not Detected	Not Detected	Not Detected

Appendix II

1. Accessibility Feature

Windows offers Accessibility Features that aim to help the disabled or those that have a unique work style or preferences. The features include an on-screen keyboard, magnifier, narrator, display switcher and an app switch. Each feature has a file associated with that specific function. These features can be launched with a specific key combination even before a machine authenticates. Attackers either replace the Accessibility Features file to launch another application or update the registry to point to an executable other than the legitimate file. The attacker can then execute these keystrokes on a physical machine or via the Remote Desktop Protocol. For instance, replacing the sethc.exe file with the cmd.exe file will allow the attacker to press the shift key five times at the login screen, and the cmd.exe file (command prompt) will launch before the user has authenticated. The command prompt has system privileges and can reset a password or execute other malicious tasks.

2. APPCert DLL

The AppCert_DLLs infrastructure provides an easy way to hook system APIs by allowing custom DLLs to be loaded into the address space of every interactive application (AppInit DLLs, 2018). Both legit applications and malicious code could utilize AppInit DLLs to hook application programming interfaces. This process hooks to the standard system interfaces or it implements unknown functionality. The AppCert DLLs are listed in the following registry key: HKEY_LOCAL_MACHINE \System\ControlSet\Control\Session Manager\AppCertDLLs. Very simple programs can be created in C++ utilizing the RegCreateKeyEx and RegSetValueEx. Attackers can create or alter AppCert DLL registry keys to cause malicious code to be executed in the context of a legitimate process to maintain persistence.

3. AppInit DLLs

This is a section of the registry that automatically loads a DLL into memory using the user32.dll. If a file is listed in this section, it loads automatically at startup. The

AppInit DLL functionality is disabled in Windows 8 and later versions when secure boot is enabled. However, attackers can easily change these settings to allow this attack vector.

4. Application Shimming

Application Shimming is functionality that is allowed by Microsoft for backwards compatibility. The Shimcache is used to determine if a module requires shimming for compatibility. In other words, the shimcache is the database that defines which applications require backwards compatibility. Windows utilizes the shimcache as a buffer between a program and the operating system (ATT&CK Matrix, n.d.). Windows created shims to run in user mode so that they cannot affect the kernel. Administrative privileges are required to install or update shims. The MITRE ATT&CK framework recognized that shims can bypass User Account Controls (UAC), inject DLLS into processes (InjectDLL), disable Data Execution Prevention (DisableNX), disable Structure Exception Handling (DisableSEH), and intercept memory addresses (GetProcAddress).

5. Authentication Package

The Local Security Authority (LSA) is the security manager for Windows systems and is responsible for local security policy and authentication. It provides support for multiple logins and more than one security protocol for the system. When a file is placed in the HKLM\SYSTEM\CurrentControlSet\Control\Lsa\ with a value of "Authentication Package"=, the binary executes as an authentication package. Mitigation of the issue with "Authentication Package" occurs by requiring a signed certificate by Microsoft for Windows 8.1 and later.

6. BITS Jobs

Windows Background Intelligent Transfer Service (BITS), as its name implies, manages background transfers such as updaters and messenger services. These applications operate in the background without interfering with other networking services. Adversaries can utilize BITS jobs to download and execute malicious code.

7. Bootkit

Bootkits alter the system and the master boot record (MBR) of the hard drive with malicious content. Because the MBR loads before the operating system, a bootkit makes

it difficult to perform full remediation unless the incident response team knows where to look for the issue.

8. Change Default File Association

In a Windows system, file association is a registry setting that defines what program opens when a file is clicked based on the file's extension. These values can be updated by the user or by a program. Attackers can utilize this functionality to change file extension information and cause arbitrary code to run. The arbitrary code executes and then the normal program associated with the file extension runs so as to not cause suspicion from the user.

9. Create Account

In order to maintain persistence, an attacker may try to create a local or domain account. If the account can be created and unnoticed, the attacker has the ability to persist in the environment until the account is detected and removed. In a domain environment where hundreds to thousands of accounts exist it may be very easy to create an account that goes unnoticed. Other than the malicious account, there are no other artifacts left behind to detect the perpetrator.

10. DLL Search Order Hijacking

Windows has a search order that is applied when executing applications. The application searches the defined path for a DLL that it requires to execute. This procedure places the DLL in a file location that can be utilized by the system before the legitimate DLL. There are other possibilities for this technique to utilize this weakness. However, the test that was conducted copied a program and its DLL to another folder to execute. Since both files are in the same folder, it relies on the new location and DLL to function. The DLL can then be changed to add malicious content.

11. Hidden Files and Directories

This procedure hides files and folders in a logical way to conceal malicious activity from the user. However, unlike the other techniques, this functionality cannot be effective by itself. The adversary must use another technique to take advantage of hidden

files and folders. For instance, a registry change must be made in order to execute the hidden file.

12. Hooking

Hooking allows adversaries to load malicious executables within the context of another process. This hides the execution allowing the injected process the memory space and privileges of the imitated process. Three potential hooking techniques include: hooking procedures, import address table (IAT) hooking and inline hooking. Hooking is used by rootkits to hide the presence of a file or the existence of a process when commands are run to list the files or processes.

13. Image File Execution Option Injection

This functionality allows a developer to attach a debugger to an application. This, in turn, causes the debugger application to launch when the intended program is executed. This is done by adding a registry entry that defines the Image File Execution Option (IFEO) with its associated program and debug program. The MITRE ATT&CK site provides examples of code that an attacker can actually use to exploit this vulnerability. Attackers can utilize IFEO to maintain persistence and to elevate privileges by executing malicious code as a debugger.

14. Logon Scripts

Windows login script offers a persistent mechanism for attackers to run specific code each time a user logs into a machine. Login scripts can perform administrative tasks. A savvy attacker might update a domain login script that will execute for a large number of users. Updating a domain login script requires domain administrative access.

15. LSASS Driver

The Local Security Authority (LSA) is the security manager for Windows systems and is responsible for local security policy and authentication. The Mimikatz tool exploits a vulnerability around the LSA Subsystem Service (LSASS) by identifying passwords in clear text found in the subsystem. Also, adversaries can replace or add malicious drivers to the LSASS process that achieves arbitrary code execution.

16. Modify Existing Service

Attackers can modify existing Windows service configurations including a path to the service's binary. Service information is stored in the registry and can be easily updated. The sc.exe, Reg.exe or custom utilities can be utilized to make changes to existing services. Malicious content can be replaced and every time the machine boots the malicious content will persist on the machine.

17. NetSH Helper DLL

Netshell is a tool that interacts with the network configuration of a machine. It is a command-line tool and can include scripting. It includes helper DLLs to extend the functionality of the program. The helper DLLs are defined in the registry. The cyber criminals can either create a new netsh.exe file or create new helper DLLs.

18. New Service

New Windows services are similar to the modify services section above. Service configuration information is stored and loaded from the registry. An adversary needs to be able to create a new service to maintain persistence. Making registry changes requires administrative privileges.

19. Path Interception

The path is a structure within Windows that provides the order in which to look for executable files. Path Interception occurs when the malicious content is found in the path before the intended content.

20. Port Monitoring

Port monitoring sets a malicious DLL to be loaded by either printer spooler service or spoolsv.exe. The spoolsv.exe runs with SYSTEM level privileges. This is accomplished via registry keys that contain entries for Local Port, Standard TCP/IP Port, USB Monitor or WSD Port.

21. PowerShell Profile

The PowerShell profile is a script that runs upon execution of PowerShell that customizes the user's environment. Different host programs utilize the profile differently.

For instance, PowerShell console, PowerShell ISE and Visual Studio Code could all have different profiles. In a domain environment, administrators can configure profiles that pertain to all Windows endpoints. Adversaries utilize this functionality to add malicious functionality to be executed upon PowerShell initiation. The NoProfile (preceded by dash) execution attribute is the only way that the malicious code would not run when PowerShell starts up. There are scenarios where a Help Desk technician or administrator launches PowerShell on a non-administrative system with malicious content included in the profile that could escalate privileges for the attacker.

22. Redundant Access

The redundant access section is highlighting the fact that attacker maintains multiple avenues to retain access to compromised systems. These are not new techniques but utilizing multiple different techniques that are listed under the persistence tactic. Motivated savvy attackers gain access and then do everything they can to maintain that access. This can include creating valid access accounts, VPN tunnels and/or other backdoors to ensure that access does not get cut.

23. Registry Run Keys / Startup Folder

There are multiple registry keys that execute each time a machine initializes. Attackers can create or alter these keys to cause malicious code to run each time the machine gets powered up. This includes not only the Run and RunOnce keys but also includes Explorer and RunServices keys as well. A plethora of keys exist that run at startup and can maintain persistence. Attackers can use masquerading or slight change to the old entry to make it appear that the registry keys associate to the correct program. This could include changing a character such as “O” to a zero.

24. Scheduled Tasks

Windows Task Scheduler can be utilized to schedule a program or script to run at a specific date and time. This allows an attacker to maintain persistence on a machine by causing their malicious code to be run again after a reboot. Task scheduling can be scheduled on a remote system if the person scheduling the task has administrative rights and file and printer sharing is enabled.

25. Screensaver

According to MITRE's website, screensavers are Portable Executables (PE) that are configured to run after a configurable time of inactivity. They are stored in the C:\Windows\System32 and C:\Windows\SysWOW64 folders. Screensaver configurations are stored in the registry. Both of these options create avenues for the adversary to utilize screensavers for malicious intent. They can replace a legitimate screen saver with an executable or they can set the screensaver settings in the registry to malicious content. During testing, it was noted that you can set the screensaver registry settings to initiate cmd.exe file and the command prompt pops up on the screen after the timeout period.

26. Security Support Provider

Windows Security Support Providers (SSP) DLLS are loaded at startup into the Local Security Authority (LSA) process. The SSP DLLs after loading into the LSA have access to security credentials such as encrypted and plaintext passwords. The registry stores these SSP configurations. The adversary can modify these entries in the registry to gain access to security credentials and these modifications persist through system reboots.

27. Service Registry Permission Weakness

Service configurations are stored within the Windows Registry. The adversary can check service registry permissions to identify weaknesses. If permission weaknesses exist, the adversary can change the executable by changing the ImagePath or binPath parameter through either the sc.exe, PowerShell, Regedit or Reg tools. This weakness can offer the attacker another persistence mechanism to run malicious code. The security context of the compromised service determines what rights are allowed to the adversary's executable such as SYSTEM privileges.

28. Shortcut Modification

Shortcuts and symbolic links provide connections to files or program when clicked or started via autorun functionality. Attackers utilize these connections to run malicious code for persistence. Shortcut modification hides what is happening when a button is clicked and therefore, can masquerade the malicious code to look like the compromised code. Another method of shortcut modification is to change the icon of a

program to point to a malicious server on the Internet. Utilizing this method, the malicious server running an SMB service will request in the background the username and password of the machine that is loading the icon. This allows the attacker to constantly be updated with the user's login credentials on a real-time basis.

29. Windows Management Instrumentation Event Subscription

Windows Management Instrumentation (WMI) allows management of Microsoft Windows personal workstations and servers through scripting languages such as VBScript and PowerShell. WMI allows administrators to manage remote systems. Attackers can utilize WMI to subscribe to events such as the wall clock time or the computer's uptime which will cause the event to execute arbitrary code.

30. Winlogon Helper DLL

Winlogon is responsible for logon/logoff and secure attention sequence (SAS). SAS is triggered by the Ctrl-Alt-Delete key combination. Registry keys are used to manage additional helper executables and other supplemental code. Malicious modifications to these Registry keys may cause Winlogon to load and execute malicious DLLs and/or executables. The three subkey weaknesses under Winlogon are Notify (notification), Userinit (userinit.exe), and Shell (explorer.exe). These subkeys can be used for malicious intent by repeatedly executing code and persistence.