A framework towards fusing multisensory cyber security data utilizing graph databases

Lori Murray
Iowa State University

A framework towards fusing multisensory cyber security data utilizing graph databases

by

Lori Murray

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering (Secure and Reliable Computing)

Program of Study Committee:
Doug Jacobson, Major Professor
Thomas Daniels
Yong Guan
Sigurdur Olaffson
Zhu Zhang

Iowa State University

Ames, Iowa

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Doug Jacobson and my committee members, Dr. Thomas Daniels, Dr. Yong Guan, Dr. Sigurdur Olaffson, Dr. Zhu Zhang for their guidance and support throughout the course of this research.

In addition, I would also like to thank my friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience. I want to also offer my appreciation to those who were willing to take part in my surveys and observations, without whom, this thesis would not have been possible.

ABSTRACT

Current network monitoring technologies do not keep up with increasing size and complexity of log data being monitored due to the ever-quickening evolution of adversary tactics. Network monitoring architectures and tactics must adapt to accommodate the increasing complexities and volumes of network data. Efficiencies can be realized by using graph databases to fuse data from the increasing data sources by generating network graphs modeling host behaviors while preserving relationships of hosts behaviors across various locations in a network. Challenges to solving this problem are fusing relevant data to construct the network graph when working with data requiring intensive relationship handling and defining the data structure of the network graph given the end goal of applying analytics. This dissertation supplies a framework to fuse data from multiple security log sources utilizing graph databases.

CHAPTER 1. OVERVIEW

Introduction

Current network monitoring technologies do not keep up with increasing size and complexity of log data being monitored due to evolution of adversary tactics and resource limitations for correlating single events across complex networks. Network monitoring architectures and tactics used by Security Operations Centers (SOCs) must adapt to accommodate the increasing complexities and volumes of network data. Efficiencies can be realized by using graph databases to fuse data from the increasing data sources by generating network graphs modeling host behaviors while preserving relationships of hosts behaviors across various locations in a network.

Network monitoring tools such as Security Information Event Managers (SIEM) ingest network log source events to correlate then alarm when a predefined rule-based signatures. These technologies use relational or nonrelational databases to organize events for correlation. To maximize efficiency SIEMs and other tools are being integrated with Security Orchestration, Automation, and Response (SOA&R) [1] [2] tools to reduce human time spent on incident response. Relational and no SQL databases fail to efficiently scale to model the highly interrelationships of the ever-increasing complexities of log data types and volumes [2]. Integration of added tools to monitoring new data source types further overloads security analysts and cyber threat hunters.

Data type log sources include not only every endpoint or network monitoring log source, but also threat intel data documenting threat actor tactics, techniques, and procedures. Adversary attacks evolve to better mask the compromise of a system to evade detection. This drives the inclusion of added data types to improve detection for system compromise. Network defense tools for monitoring enterprise traffic heavily rely on real time detection capabilities. With proper storage of network traffic data, indicators of compromise are generated based on detected and confirmed network attack. These indicators of compromise provide the attack vector architecture which, if applied correctly to network traffic in a data storage environment, provide a tool for identifying other hosts on the network compromised with the same attack vector.

Motivation of Problem

During an investigation an incident responder gathers forensic evidence to confirm a compromise occurred. Forensic evidence resulting from an incident response investigation determines the anatomy of the attack, used to generate rule-based signatures to detect future occurrences of the same compromise. The forensic evidence provides input to the generation of the network graph by identifying what data sources are relevant to the network compromise, how the data sources are related in the context of the network compromise. Both the relevant data sources and relationships build a graph model used to generate the network graph. We generate the attack template by tailoring the graph model with detected IoC in the forensic evidence. Using the same evidence can be used for a cyber threat hunting exercise, the goal being detection of other hosts on the network impacted by the confirmed compromise from historical but recent data. *Figure 1* supplies a procedure for identifying impacted hosts.

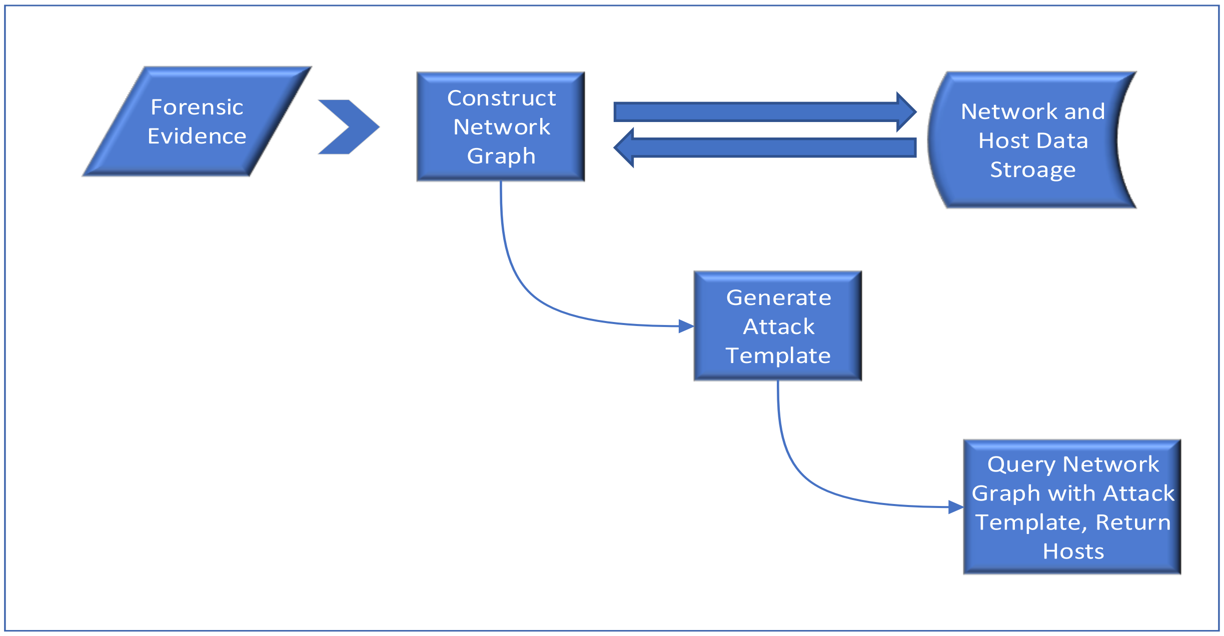

Forensic Evidence
Construct Network Graph
Network and Host Data Stroage
Generate Attack Template
Query Network Graph with Attack Template, Return Hosts

Figure 1 Procedure to Identify Impacted Hosts

Using the forensic evidence to find relevant data sources to use when generating a network graph. The attack template is configured according to the initial forensic evidence, a query consisting of IoC parameters from the anatomy of the network compromise. The last step is to query the network graph using the attack template, returning information enough to document in an incident. Enough information includes what event is detected, where the event occurred such as hostname, who performed the event such as username logged into the hostname, when the event occurred such as timestamp, and finally how the event is executed.

Challenges to Solving Problem

Analysis of network behavior pattern discovery goes beyond correlation of specific events, but across a both per user and per host basis, extending the evaluation beyond sequences of events to a more inclusive evaluating sequences of events across a per host and time basis. Constructing the network graph using a multimodal graph with groups of nodes are various parts of the attack. Multimodal graphs are heterogeneous graphs where each node belongs to a type (label) of nodes. Specifically, insights to what the sequence of events about a network compromise as well answer the where, when, how, and who in the network compromise. Where the compromise occurred by identifying impacted hosts, when the compromise happened through timestamps, who by identifying the impacted usernames involved in the network compromise, and finally how the network compromise was executed.

Two challenges impacting the procedure outlined in *Figure 1*breaks down the data mining challenge and the data modeling challenges.

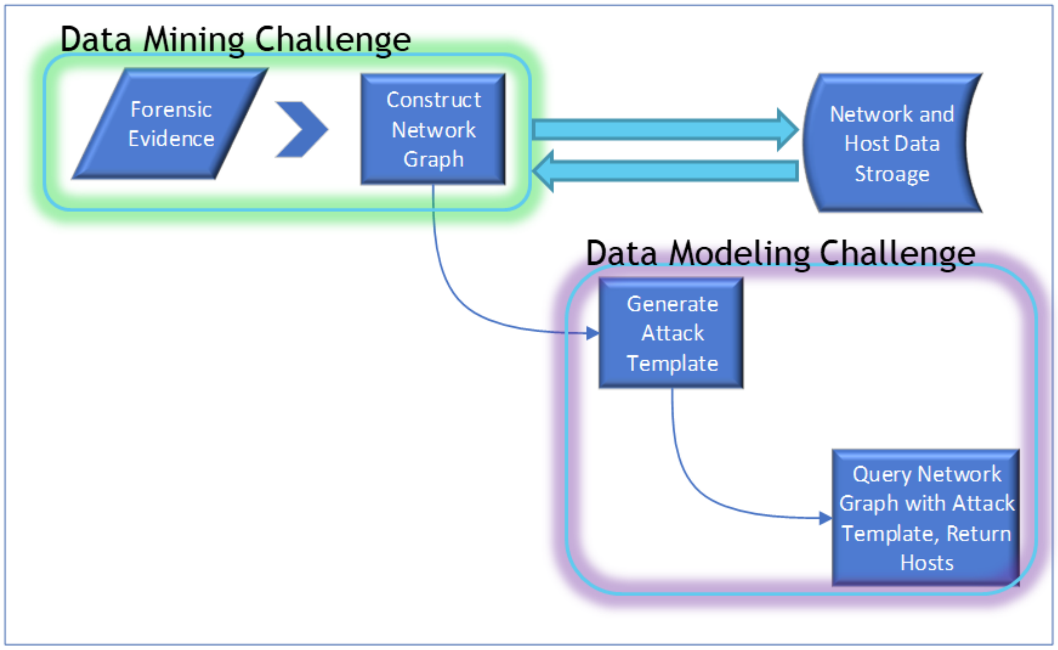


Figure 2 Procedural Challenges

The data mining challenge results from uniqueness of the data sources in each network makes it difficult to normalize classifying data sources as relevant or not relevant. Fusing relevant data to construct the network graph when working with data requiring intensive relationship handling. The network graph must model the relationships between each host node in a manner that shows the how the network compromise executes. The increasing complexity and volumes of data sources results in a data lake effect where stored data sources include structured, nan-structured, and even raw data. Somehow relevant data sources need to be fused to properly model highly related and connected dataset of network and host-based data sources.

The data modeling challenge is defining data structures of the network graph given the end goal to enable analytics algorithms. Further, applying deep learning algorithms to such graphs is limiting given the difficulty of finding and extracting relevant data from the entirety of the edge or node labels. Graphs are a type on Non-Euclidean data unlike images, text, audio. The challenge is expressing graphs computationally enabling application of graph analysis and graph learning, this is done through generation of adjacency and incidence matrices, a degree matrix, or Laplacian matrix. [3]

Problem Statement

There is no consistent framework that organizes and fuses the diverse data sources in a manner from which actionable intelligence can be identified.

Thesis Statement

This dissertation proposes a framework addressing both the data mining and data modeling challenges achieving fusing data sources using graph databases in an organized manner from which actionable intelligence can be extracted.

Significance of Problem

In response to the increasing complexity and volumes of network data that must be monitored, Security Operations Center architecture is evolving to store more log source types and volumes [4] and move utilities performing analytics and correlation closer to where data is stored. Data type log sources include not only every endpoint or network monitoring log source, but also threat intel data documenting threat actor tactics, techniques, and procedures.

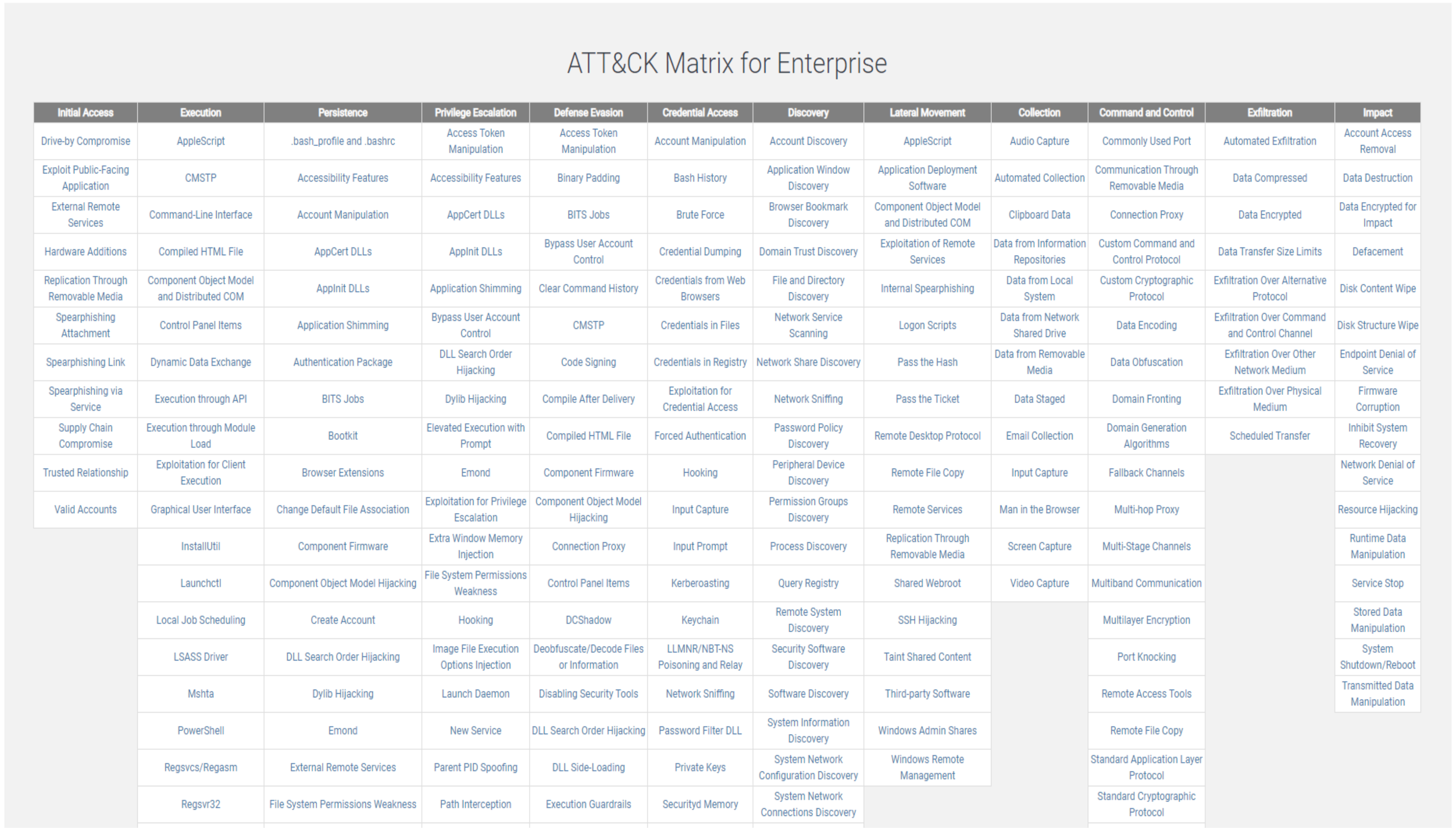
Adversary attacks evolve to better mask the compromise of a system to evade detection. This drives the increasing inclusion of more data types to improve detection for system compromise. Network defense tools for monitoring enterprise traffic heavily rely on real time detection capabilities.

With proper storage of network traffic data, indicators of compromise are generated based on detected and confirmed network attack. These indicators of compromise provide the attack vector architecture which, if applied correctly to network traffic in a data storage environment, provide a tool for finding other hosts on the network compromised with the same attack vector.

This dissertation focuses on how to prepare data for ingestion to enable fusing across several disparate data source types and a generalized network graph model to support the incident response investigation use case of cyber threat hunting.

Background Information

With the constant evolution of adversary tactics used during attacks, computer network defense teams have shifted from relying on signature based tools using predefined Indicators 0f Compromise (IoC) to more advanced analysis procedures using behavioral detection analytics techniques. [5] Figure 3 shows the MITRE's ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) comprehensive matrix of adversary tactics, techniques and procedures (TTPs) used to classify attacks for threat modeling activities. The TTPs are mapped back to specific threat actors, describing both host and network-based events saw during each cyber-attack. Since being released, the MITRE ATT&CK framework enabled network defenders, cyber threat hunters, and researchers to perform analytics, starting a transition from reactive monitoring using Indicators of Compromise (IoC) to more proactive defense using TTPs.



ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
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	Accessibility Features	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	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Component Object Model and Distributed COM	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	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	Internal Spearphishing	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Content Wipe
Spearphishing Attachment	Control Panel Items	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Disk Structure Wipe
Spearphishing Link	Dynamic Data Exchange	Authentication Package	DLL Search Order Hijacking	Code Signing	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Endpoint Denial of Service
Spearphishing via Service	Execution through API	BITS Jobs	Dylib Hijacking	Compile After Delivery	Exploitation for Credential Access	Network Sniffing	Pass the Ticket	Data Staged	Domain Fronting	Exfiltration Over Physical Medium	Firmware Corruption
Supply Chain Compromise	Execution through Module Load	Bootkit	Elevated Execution with Prompt	Compiled HTML File	Forced Authentication	Password Policy Discovery	Remote Desktop Protocol	Email Collection	Domain Generation Algorithms	Scheduled Transfer	Inhibit System Recovery
Trusted Relationship	Exploitation for Client Execution	Browser Extensions	Emond	Component Firmware	Hooking	Peripheral Device Discovery	Remote File Copy	Input Capture	Fallback Channels		Network Denial of Service
Valid Accounts	Graphical User Interface	Change Default File Association	Exploitation for Privilege Escalation	Component Object Model Hijacking	Input Capture	Permission Groups Discovery	Remote Services	Man in the Browser	Multi-hop Proxy		Resource Hijacking
	InstallUtil	Component Firmware	Extra Window Memory Injection	Connection Proxy	Input Prompt	Process Discovery	Replication Through Removable Media	Screen Capture	Multi-Stage Channels		Runtime Data Manipulation
	Launchctl	Component Object Model Hijacking	File System Permissions Weakness	Control Panel Items	Kerberoasting	Query Registry	Shared Webroot	Video Capture	Multiband Communication		Service Stop
	Local Job Scheduling	Create Account	Hooking	DCShadow	Keychain	Remote System Discovery	SSH Hijacking		Multilayer Encryption		Stored Data Manipulation
	LSASS Driver	DLL Search Order Hijacking	Image File Execution Options Injection	Deobfuscate/Decode Files or Information	LLMNR/NBT-NS Poisoning and Relay	Security Software Discovery	Taint Shared Content		Port Knocking		System Shutdown/Reboot
	Mshta	Dylib Hijacking	Launch Daemon	Disabling Security Tools	Network Sniffing	Software Discovery	Third-party Software		Remote Access Tools		Transmitted Data Manipulation
	PowerShell	Emond	New Service	DLL Search Order Hijacking	Password Filter DLL	System Information Discovery	Windows Admin Shares		Remote File Copy		
	Regsvcs/Regasm	External Remote Services	Parent PID Spoofing	DLL Side-Loading	Private Keys	System Network Configuration Discovery	Windows Remote Management		Standard Application Layer Protocol		
	Regsvr32	File System Permissions Weakness	Path Interception	Execution Guardrails	Securityd Memory	System Network Connections Discovery			Standard Cryptographic Protocol		

Figure 3 MITRE ATT&CK Framework

Adversary TTPs constantly evolve to mask system compromises obfuscating network detection tools, anything from generating duplicate processes on a host or using unpatched embedded devices as a pivot point. The effect of this is inclusion of host and network data types as well as new data types that historically are not checked.

For example, Mirai, an internet of things botnet infected over 600,000 devices at its peak caused distributed Denial of service attacks. The self-propagating worm actually succeeded by using default credentials to log into IoT devices, installing malware to establish communications with a Command and Control (C2) server. [6] The Mirai infection is an example of a device not normally monitored in an enterprise network as an embedded device. If an enterprise begins checking of embedded or IoT devices, this adds both volume and complexities to network monitoring. As adversaries find new ways to obfuscate monitoring tools such as Security Information and Event Monitoring (SIEMs), adding more data types to the monitoring space. The effect is a vast increase to both the volume of data that must be watched and complexities to the types of data being checked.

CHAPTER 2. LITERARY REVIEW

Searching for current literature for this thesis topic bridges two very different fields of research, thus the search much also be performed from both perspectives, either from a cyber security perspective of using machine learning for network monitoring and cyber hunting or from the perspective of a data scientist working on network monitoring. Oddly, each search resulted in vastly different resources with little overlap. This section reviews current literary available for three key topics related to this dissertation:

1. current applications of machine learning or analytics to data sets to answer cyber security questions,
2. use of graph databases to model security use cases, and
3. measuring similarity in graph databases.

Current Applications of Machine Learning to Cyber Datasets

Tools utilized for either detecting a host compromise or network attack using machine learning is very dependent on the correlation of information events, defining a behavioral baseline then applying machine learning to detect anomalies or patterns [2]. Such techniques result in a high false positive rate and large monitoring gaps [7]. Improving such challenges require enhancing signatures used for monitoring with supplemental data that is updated in real time or utilize such tools for cyber hunting [2]. A network based cyber hunter detects anomalies in the network using tools that return information from which insight can be pulled [8] [2]. For example, statistical analysis can be used to find uncommonly used communication protocols on a network [9]. Given the rise in this type of activities drives storage of ever-increasing network data sets stored in a warm or cold storage and tools for evaluating these datasets [2].

Applying Graph Databases to Model Security Use Cases

Utilizing machine learning for network monitoring should have a focused problem detection with a well-defined target of analysis to avoid high false positive rate. For example, tools that simply detect anomalies on an entire network must be taken with a grain of salt. Research focuses more and more on detecting very specific events of compromise, either host [10] or network [7]. In doing so a target of analysis is defined (host or network), threat vector to be checked (anomalous behavior, email, file transfer) along with an observed change of state. This gives rise to the notion of attack vector graphs [11] [12] [13] to model relationships in a network node.

Measuring Similarity in Graph Databases

Attack vector graphs use graph databases to stand for the relationship between hosts as nodes and the relationship as edges. Assigning weights to the edges reflect differences in edges of the same relationship [13] [14]. For example, for time periodicity a higher number is larger time intervals. Graph databases are used to align network attack behaviors with source threat actors [14]. Graph databases can model both network actions to generate a behavioral indicator of compromise. Cyber threat intelligence traces different network attack behaviors to a threat actor group, common to clustering malware families based on the behaviors [15]. The combination of a network attack vector with threat actor profiles allow for network traffic to be attributed based on common behaviors of the network attack [14] [15].

Similarities between differing graph database models can be measured through evaluating the substructure of each graph database model [16], potentially returning a measure of similarity. Measuring similarity between graph databases has been applied to other topics of research sub as brain connectivity networks [17] or detecting anomalies in elective networks of a smart grid. [18]

CHAPTER 3. DATA MINING CHALLENGE

The ever-increasing volumes of data and number of data types being watched started the data lake concept as SOC architectures adjust to accommodate increasing complexity and amounts of data. A data lake is a centralized repository storing vast amounts of structured and unstructured data. *Figure 1 Procedure to Identify Impacted Hosts* documents a procedure where data taken from a data storage for mining of patterns specific to a detected network attack. Using the Cross-Industry Standard Process for Data Mining (CRISP DM) [19] shown in Figure 4 with a few modifications we discuss a framework for approaching data preparation for generating a network graph.

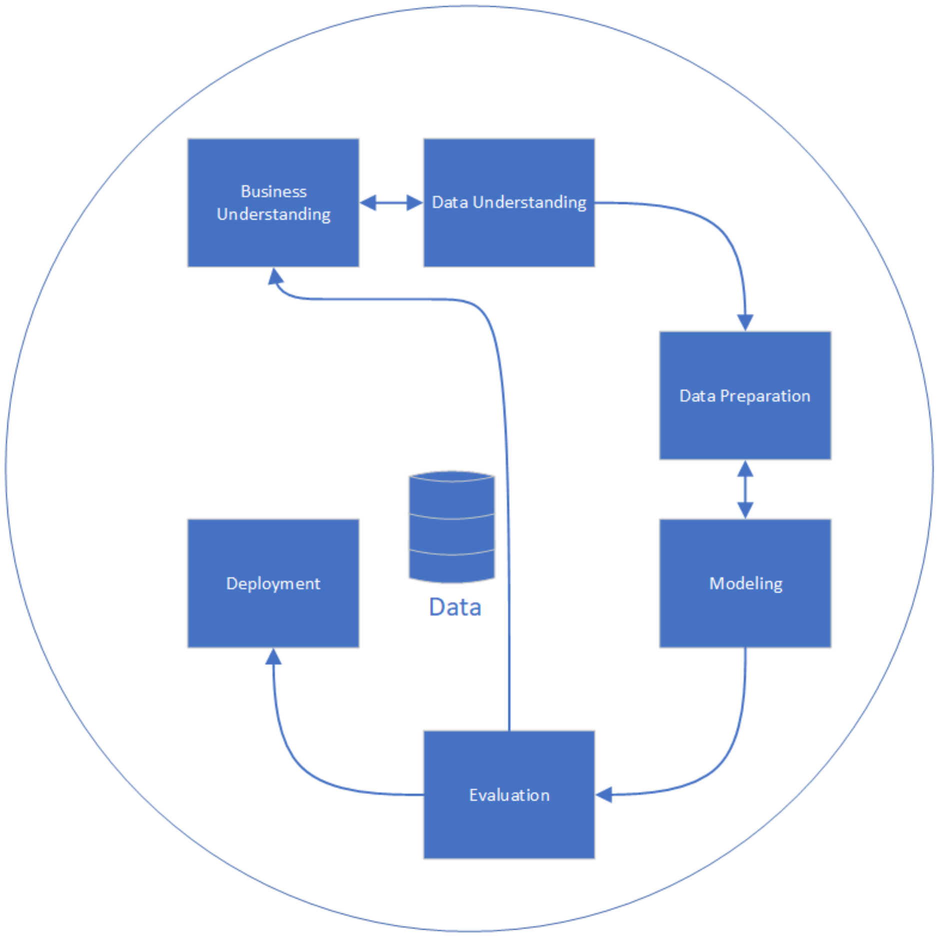


Figure 4 Cross-Industry Standard Process for Data Mining

CRISP-DM breaks down the data mining process into the following steps [20].

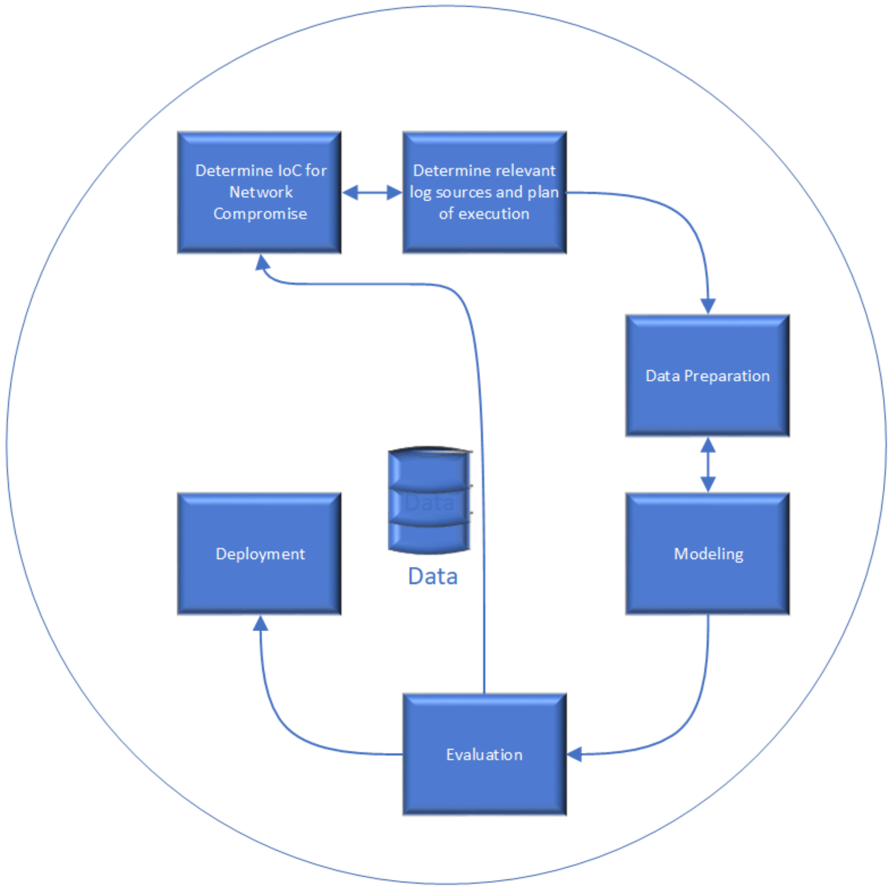

Determine IoC for Network Compromise
Determine relevant log sources and plan of execution
Data Preparation
Modeling
Evaluation
Deployment
Data

Figure 5 CRISP-DM for Generating Network Graph

Applying Business Understanding

For CRISP-DM this step focuses on understanding the project goals and requirements from a business perspective, and then converting this knowledge into a data mining problem definition and a preliminary plan.

Applying Business Understanding to cyber threat hunting it is the step of taking the forensic evidence from an incidence response investigation to determine, given the sequence of events during the network compromise, what data sources log the activity. The output of this is what indicators of compromise are most relevant determining relevant log sources.

Data Understanding and Preparation

Starts with first data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.

Understanding the log source data includes examining the characteristics of the log sources including determining if the data is structured or not structured, data type. Specifically, if the data type is not categorical or numerical, what needs to happen to the data to vectorize the fields for analysis. For example, data attributes that are strings or text need to be quantified. These goals need to be defined prior to generating a network graph model and can done using algorithms such as similarity measurements.

Modeling and Evaluation

Modeling techniques are selected and applied. Since some techniques like neural nets have specific requirements regarding the form of the data, there can be a loop back here to data prep.

Again, very similar between the core CRISP-DM and the purpose of this use case, including understanding the requirements of the algorithms and techniques such as neural nets or graph learning which need to be applied to the network graph. Once one or more models have been built that appear to have high quality based on whichever loss functions have been selected, these need to be tested to ensure they generalize against unseen data and that all key business issues have been sufficiently considered. The result is the selection of the champion model(s).

Deployment

Generally, this will mean deploying a code representation of the model into an operating system to score or categorize new unseen data as it arises and to create a mechanism for the use of that new information in the solution of the original business problem. Importantly, the code representation must also include all the data prep steps leading up to modeling so that the model will treat new raw data in the same manner as during model development.

The CRISP-DM procedure can be directly applied for preparing data to generate a network graph step outlined in *Figure 1.* Additionally, the procedure supplies input to the actual generation of a network graph.

CHAPTER 4. DATA MODELING CHALLENGE

There are several use cases which graph databases can be used to more efficiently perform investigations in a network. The network model used to fuse the source data using graph databases is dependent on the investigative question under evaluation. For example, a network model to monitor risk associated with vulnerability management will look different than a network model used to perform incident response investigations. In the latter, the network model must incorporate network infrastructure with network security posture. Examples of network security posture include infrastructure configuration, patch level, and ports/protocols/services (PPS) that are running. The final layer of information required for this network models is latest applicable vulnerabilities [21] to build a network graph that stores information required for a situation assessment model.

A graph database is a No SQL database using a graph structure including nodes representing objects and edges representing relationships between the nodes (objects). Graph databases are more effective than other No SQL or relational databases at showing relationships in highly interconnected datasets. [22] There are distinct types of graph models. Figure 6 shows an example graph database example as the property graph model, the graph is formed of nodes, edges and properties.

- Nodes are entities in the graph labeled with the role or identity of the node. A unimodal graph has one category of nodes where multimodal graphs employ many group types of nodes.

- Edges are relationships between nodes are directed to reflect the relationship between the start and end node.
- Properties are metadata applied to either or both edges and nodes. Properties can be quantitative or qualitative, but to support graph learning data must be capable of being vectorized.

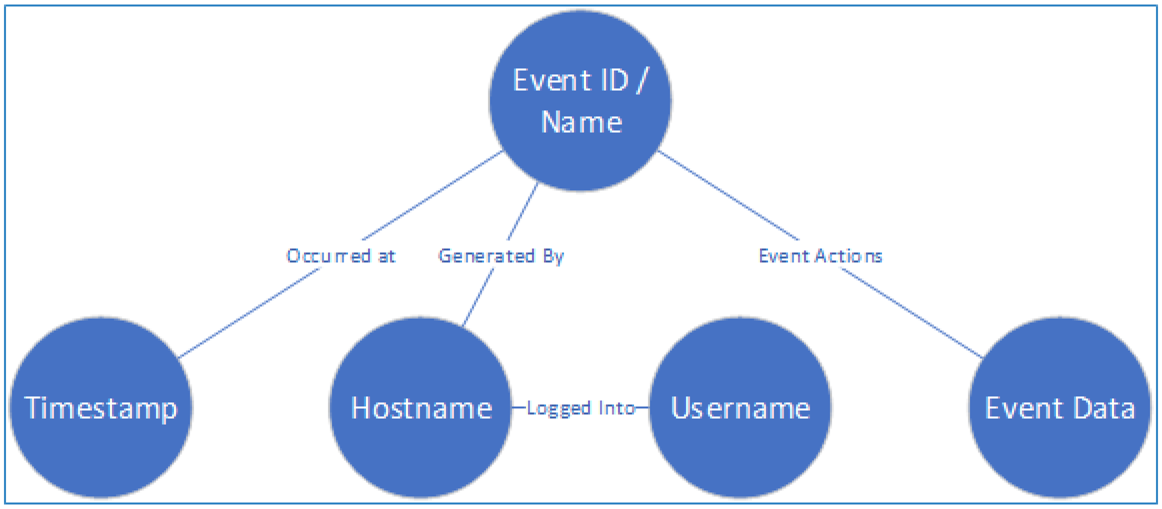


Figure 6 Example Graph Database

The network graph models the relationships of what events happen on the network, how events executed through the sequence of actions. The network graph models behaviors over both time using the event timestamp and location using the event hostname. Host IP address could also be used, but for this discussion hostname is used due to the dynamic nature of IP leases. Figure 3 shows the generalized node types along with the relationships between each. Generalized is used to describe the different node groups. The network graph models the relationship between, at a minimum, Event ID / Name, Timestamp, Hostname, and Event Data resulting in a multi-partite graph where edges exist only between nodes within each node group. Not every logged event includes Username, so the edges from Username nodes connect with Hostname nodes. Event data is a collection of actions with objects executed during the Event ID

/ Name, therefore there may be a one to many relationships from Event ID / Name to Event Data node groups.

Performing investigations require incident responders to determine the context of a network compromise. The NIST Incident Response Life Cycle details four steps of an investigation, step two performs the forensic evidence of a detected compromise to define Indicators of Compromise. Step three Containment Eradication and Recovery is where the detection of other hosts impacted the same compromise occurs [23]. The investigation determines where the compromise happens, including other hosts that may be affected. Keeping with the SOA&R trend, follow up actions include developing monitoring for future compromises and mitigation of other impacted hosts on the network.

Constructing a network graph by fusing the correct data from relevant domains provide the answers of where the compromise is located, who is involved with the compromise, what the compromise is, and how the compromise happened. Each information domain houses a group of nodes having the information shown in Figure 7. For example, Network Infrastructure Domain includes an inventory of network elements, hostnames, IP addresses (if applicable), role, as well as other relevant data. User Security Domain includes relevant account types and privilege levels in the network, sourced with Identity and Access Management data logs. The Network Activity Domain includes event logs from different network and host-based intrusion detection and management systems such as firewalls or SIEM tools.

Considering applying building a graph model for the cyber threat hunting use case the network graph needs to model:

- What events were detected
- How the detected events were executed
- Where the detected events occurred
- Who initiated the detected events?
- And, if applicable,
    - When the detected events occurred

Taking into consideration every network compromises the domains of information to combine into the network graph is unique. The domains are dependent on the original question outlined in the Business Understanding step. For example, if the question is what accounts performed which actions on the network the information domains shown in the example Figure 7. The common sectors in all will be building the model on top of the Network Infrastructure Domain. If user activity is involved, for example within lateral movement or privilege escalation, then the second domain includes user security activities. The number of domains is determined by the amount of information required. The final domain to include is the network activity logged in data sources classified as relevant.

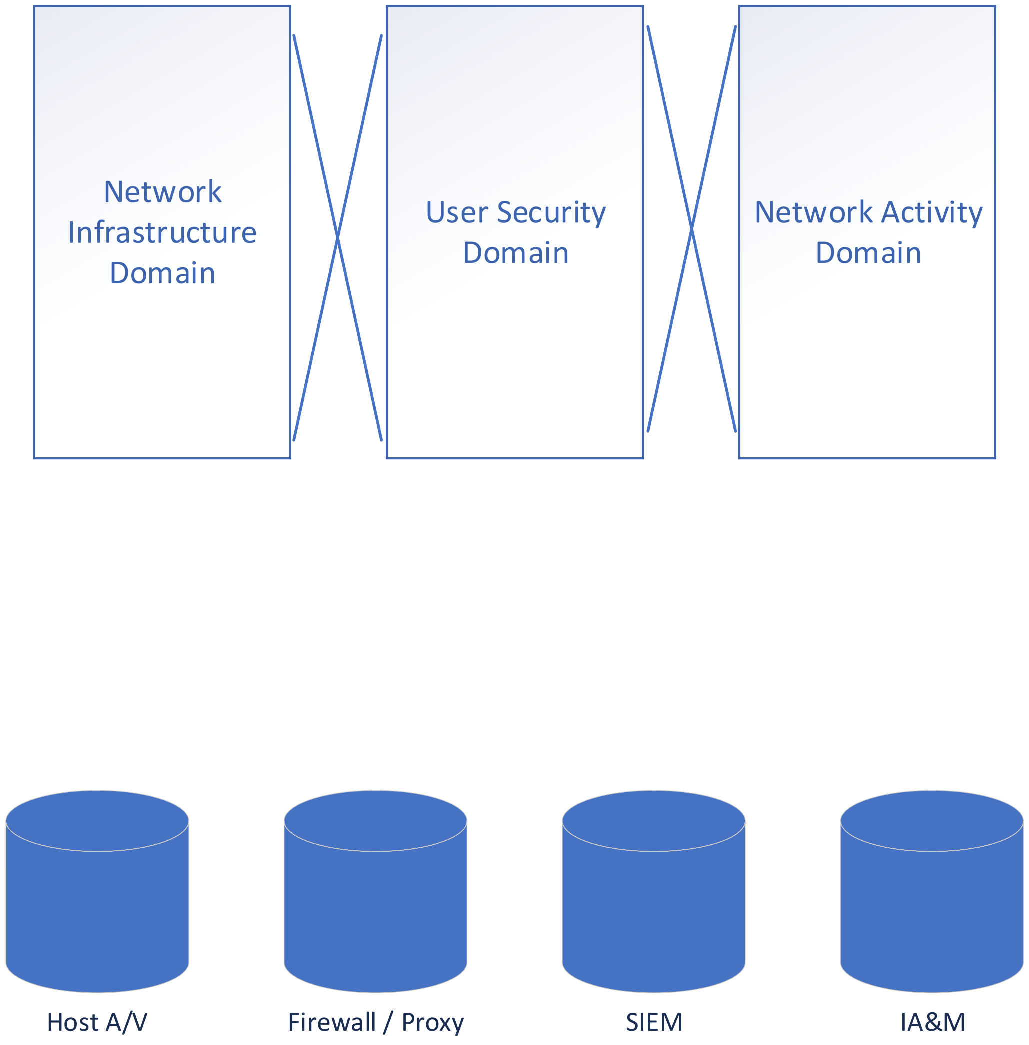


Figure 7 Example Information Domains

Beginning with the physical layer by modeling network infrastructure. Creating a node for each network entity shown in Figure 8, adding property tags to enumerate relevant profile data such as role of the network entity. Generally, applying a uniqueness constraint for each hostname node prevents duplication. Logically, network entities are not duplicated in deployment. Integrating user account activity by generating a node for each user account at the highest level of access provisioned. User Accesses property labels include provisioned access and roles with provisioning timestamps. If possible, including a list of normalized logon activity can be used for queries. Lastly, generating nodes for ingested network and host-based events generated by network monitoring tools. Indicators of Compromise determined by the forensic investigation driving the data source types to include in the network and host event input. The network and host events include event identification numbers, event names, event field data, and timestamps. More properties can be included.

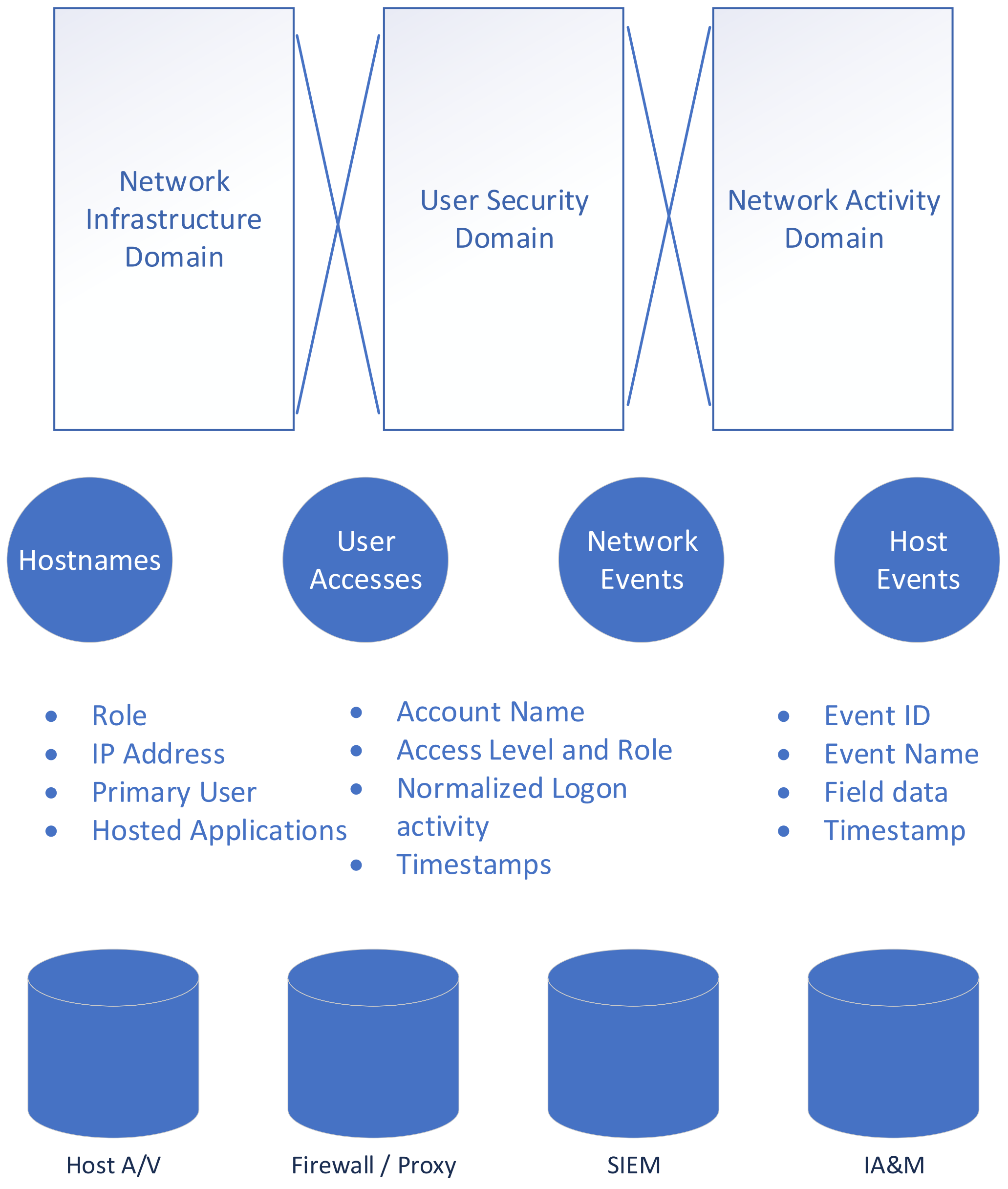


Figure 8 Example Graph Model Nodes

Next define how the nodes in the graph are related, generating relationships based to model what hostnames generated which network events with what accounts were logged in during the relevant time duration. Relationships from Hostnames to Network and Host Events showing what hosts generated each event. Relationships from User Accesses to Hostnames showing what account names were logged into each host during relevant time periods.

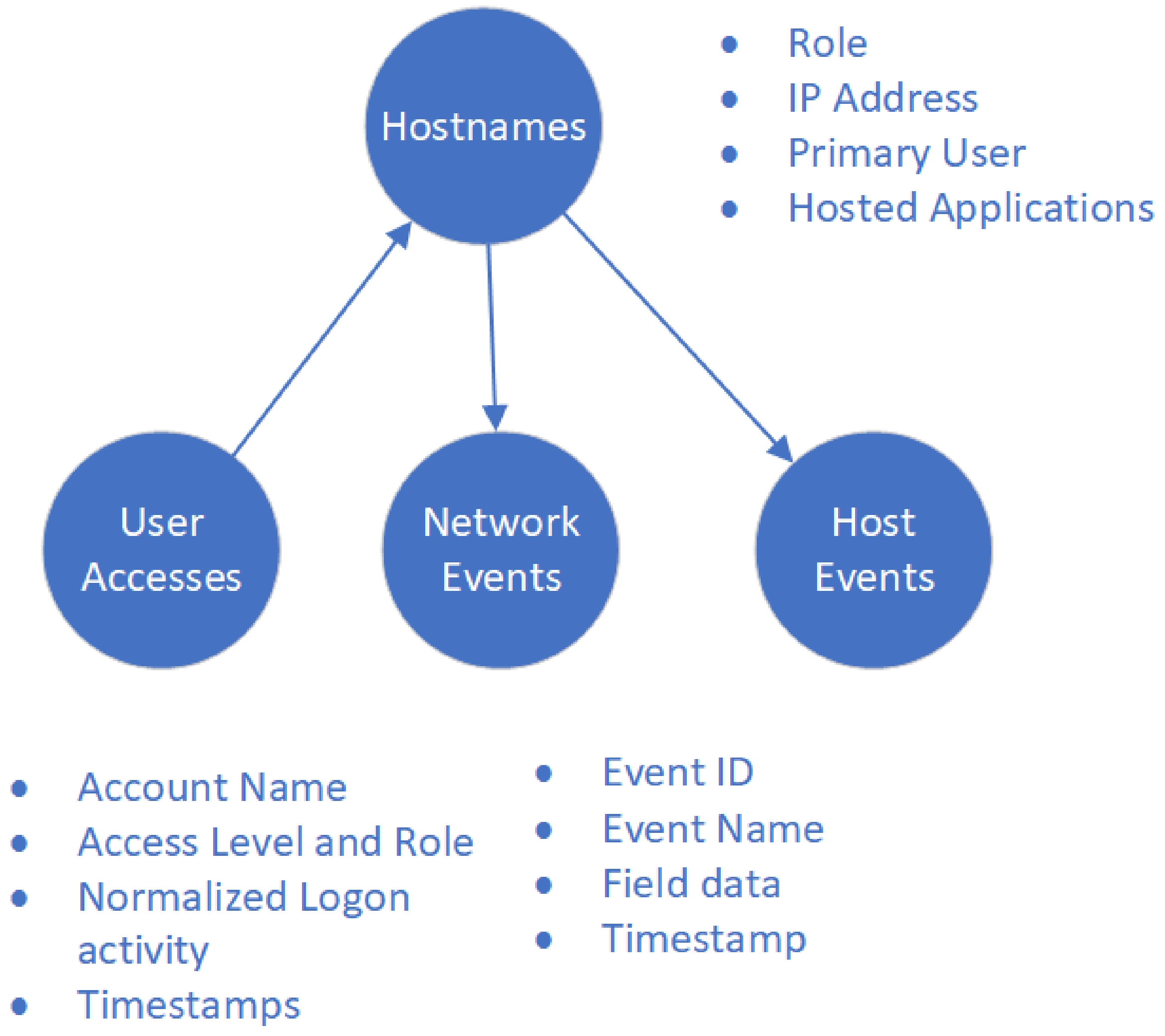


Figure 9 Network Graph Generalized Model

Requirements of Network Graph Model

Application of graph algorithms such as centrality or search tools can be used to further analyze patterns in network traffic. Constructing a network graph in a manner to model relationships not only between hosts, but also with users and event and objects used during events, enables use of graph learning algorithms for further analysis. For example, cyber threat hunters can use IoC confirmed during an incident response investigation determining if the same compromise impacts other network entities. Network defenders can utilize cyber threat intelligence (CTI) data to identify hosts impacted by a confirmed compromise, using the forensic evidence to identify other hosts that share similar behaviors using forensic evidence graphs. [24]

Data from the different log sources provide the context of what events happened, where the events happened per hostname, how the events happened by connecting the sequence of actions, who was logged into the host generating the events, and finally when the events occurred relevant determining the sequence of actions.

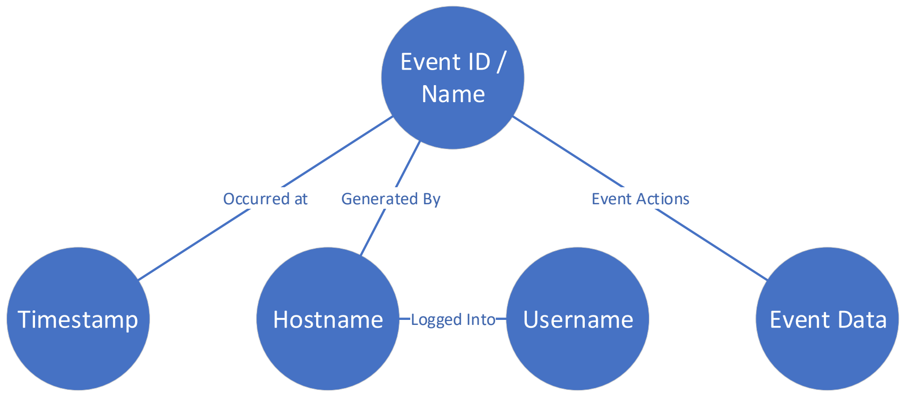


Figure 10 Example Specified Network Graph Model

Generate Attack Template

The network graph builds a model representing events so details on the who, what, where, how, and when of an incident can be extracted. The attack template begins with the forensic evidence graph from the incident response investigation. The network graph is generated using the log types found from the evidence graph. Tailoring the evidence graph with parameters found specifically in the evidence graph is used to generate a query for searching the network graph, returning output of hosts, hostnames, and usernames if applicable. For example, if adversaries gain persistence through transferring and execution of a file. Network event log sources include all sources logging file transfers in the system. Host event log sources include all sources documenting execution of files and methods the file sets up persistence such as registry key changes or process creation. Linking both sets of event nodes to hostnames a query shown in Figure 11. Generating the query to match file names or process name then traverse back to the Host then returning hostname.

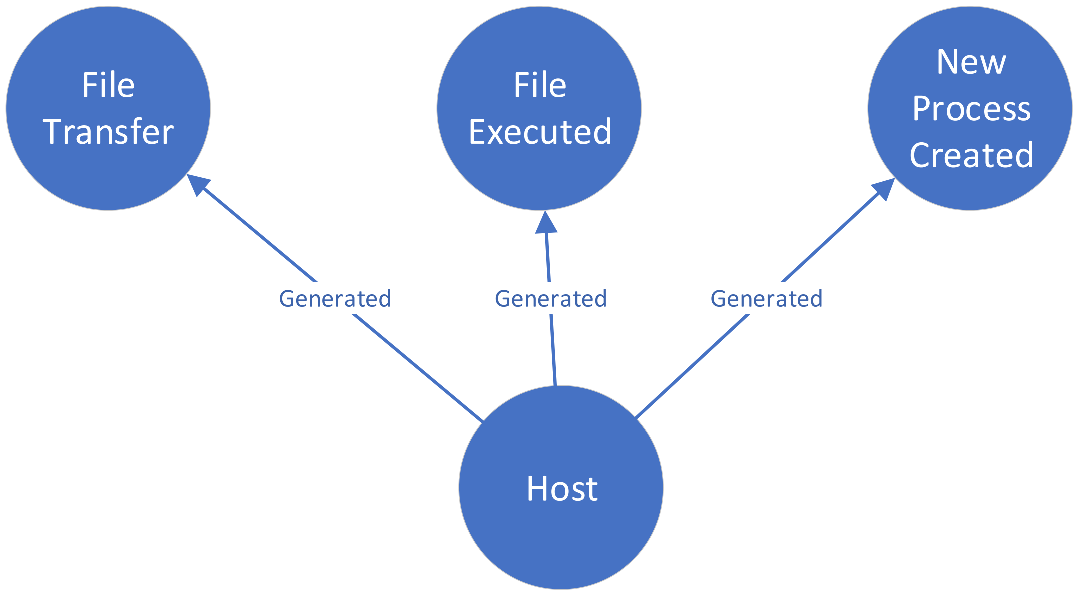


Figure 11 Attack Template Query

Expressing the Network Graph Model Computationally

The decomposition of event logs to the network graph, determining what fields in the logs are allocated to a node group and how to vectorize the fields must be considered. Computations for analysis on the network graph are optimized when the vertices in each node group are categorial or numerical data. For algorithms such as neural networks to analyze fields that are free text, such as file names or usernames, some numerical calculation needs to be identified to convert in from a text field to a numerical field.

Prior to performing analytics, the data relevant to the investigation must be identified and organized such that the algorithms can be applied returning enough information. Graph databases can be used to fuse the relevant data from the different log sources, building the network graph enabling application of graph algorithms. Eventually Graph Neural Networks (GNN). Applying GNN is not included with this analysis, however the goal of supporting this task influences the architecture of the network graph. The network graph must be constructed in such a manner it can be vectorized to serve as the input feature matrix to the GNN algorithm with the corresponding adjacency matrix.

To apply graph learning or analytics network graphs must be expressed computationally to serve as inputs to a learning or analytics algorithm. Using graph theory matrices to construct representations of the network graph while modeling the relationships as edges between graph nodes. Two network graph representations discussed below are adjacency matrices and incidence matrices.

Formally, let $G = (V_1, V_2, V_3, V_4, V_5 \ldots V_k, E)$ be a *k-partite* graph with parts for

Event ID / Name $V_1 = \{v_{11}, v_{12}, \ldots, v_{1q}\}$

Timestamp $V_2 = \{v_{21}, v_{22}, \ldots, v_{2r}\}$

Hostname $V_3 = \{v_{31}, v_{32}, \ldots, v_{3s}\}$

Username $V_4 = \{v_{41}, v_{42}, \ldots, v_{4t}\}$

Event Data from $V_5 = \{v_{51}, v_{52}, \ldots, v_{5u}\}, \ldots, V_K = \{v_{k1}, v_{k2} \ldots, v_{ky}\}$.

and Edges E include groups of edges between the vertices of each node group.

Observe: $|V_1| = q$, $|V_2| = r$, $|V_3| = s$, $|V_4| = t$, $|V_5| = u \ldots$, $|V_K| = y$; $|E| = m$.

k-Adjacency Matrix

An adjacency matrix representing the network graph is a square matrix with nodes ordered by node grouping, V_1 then V_2 then V_3 through V_K. For the defined graph G above, the k-adjacency matrix is a (q+r+s+t+u+…+y) square 0-1 matrix B in which $b_{i,j} = 1 \Leftrightarrow (m_i, n_j)$ € E encodes the edges from vertex group i to vertex group j. The adjacency matrix is constructed by ordering bi-adjacency graphs between vertex groups along the upper triangle of the matrix. For undirected graphs the adjoining adjacency matrix is symmetric, for directed graphs the adjacency matrix is not necessarily symmetric. For directed graphs the entries in the adjacency matrix are not restricted to -1 but reflects the number of nodes in links. For graphs allowing nodes to self-loop the entries on the diagonal are not necessarily 0. For construction of this example Figure 12 demonstrates the adjacency matrix for a k-partite undirected graph. To complete the symmetry of the matrix, the lower triangle of the matrix is constructed with the transpose of the bi-adjacency graphs from the upper triangle of the matrix.

$$A = \begin{pmatrix} 0 & B_{12} & B_{13} & B_{14} & \cdots & B_{1k} \\ B_{12}^{T} & 0 & B_{23} & B_{24} & \cdots & B_{2k} \\ B_{13}^{T} & B_{23}^{T} & 0 & B_{34} & \cdots & B_{3k} \\ B_{14}^{T} & B_{24}^{T} & B_{34}^{T} & 0 & \cdots & B_{4k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{1k}^{T} & B_{2k}^{T} & B_{3k}^{T} & B_{4k}^{T} & \cdots & 0 \end{pmatrix}$$

Figure 12 Constructed Adjacency Matrix

Incidence Matrix

An incidence matrix representing the network graph in this example matrix with a row per node ordered by node grouping, V_1 then V_2 then V_3 through $V_{K.}$ The incident matrix has a column for each edge type. The matrix size is (q+r+s+t+u+…+y) rows by m columns. The entries of the matrix have a 1 where an edge is connected to a node.

CHAPTER 5. USECASE EVALUATION

This section explores building network graphs and applying graph algorithms on two use case using a data set of generated network and host based traffic. This exercise demonstrates using graph databases to fuse data from different network and host data sources and how using graph algorithms to identify traffic patterns..

Use case #1: Filename Indicator of Compromise

This section examines a dataset generated through emulating APT03 TTP tactics to generate a network graph described earlier. An open source graph database tool called Neo4j is used to generate the network graph. Neo4j utilizes Cypher, a declarative graph query language allowing for efficient data querying in property graphs.

The Mordor dataset [25] generated by security researchers to enable analytics on cyber security and network attack traffic. The dataset is generated to replicate the techniques implemented in the MITRE ATT&CK Round 1 Evaluation exercise [5].   In total 38 techniques generate log traffic across ten of the ATT&CK phases. The following section documents both the data preparation procedure from Figure 4 Cross-Industry Standard Process for Data Mining then building a network graph outlined in Figure 9 Network Graph Generalized Model using the Mordor dataset using one of the emulated ATP TTPs as the initial forensic evident as defined in the MITRE ATT&CK Framework.

The Mordor dataset emulates the threat actor APT03 otherwise known as several alias terms including Gothic Panda. The generated dataset executed 56 enterprise techniques across ten ATT@CK tactics. [26] [27] The MITRE ATT@CK tactics emulated in the dataset are Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, and Exfiltration. For this implementation the evaluation focuses on the Lateral Movement phase utilizing a TTP 1105. The adversary starts lateral movement within a network using tools that copy files to remote machines. [28]

Generating a network graph focusing on one of the techniques specific to a single ATT&CK phase, with the indicator nodes pulled from source logs including information. The event names or event identification (ID) populate the properties of the What Nodes, while the hostnames populate the Who Nodes. Relevant fields in the logs become the How nodes and may include account or usernames, the object to which access is being changed, and the type of access change. The type of access change may very well include lockout and granting or removal of access.

APT03 Dataset Exploration

Prior to constructing a network graph data exploration is performed to optimize analysis to understand the types of variables in the dataset, attribute data structures to prepare for generation of the network graph. The Mordor dataset is prerecorded security events generated by simulating adversarial attacks documented in the MITRE ATT&CK Framework in a JSON format. [25] One of the intentions of the dataset creators is facilitation and enabling of advanced analytics between documented adversarial TTPs.

This section walks through the CRISP-DM Dataset Understanding and Preparation procedure to generate a network graph model with the Mordor dataset using the APT03 TTP 1105 as an attack query.

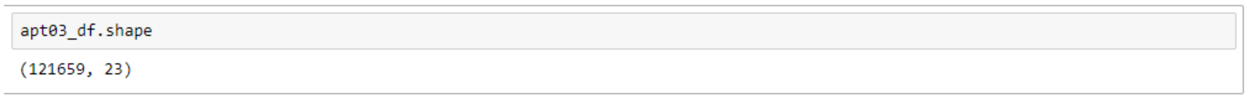
Business Understanding

This step uses the input the resulting forensic evidence from an incident response to define what IoC to search the recently historical data for. In ATT@CK Evaluation Step 16.G.1 PowerShell is used to transfer a file containing the malware stager called autoupdate.vbs to a new victim making the Microsoft Windows Security Auditing, Microsoft Windows Sysmon, and PowerShell log sources relevant as these captures executed commands. Windows Security Auditing and Windows Sysmon also log user log ins.

The generated network graph needs to show relationships between hosts on the network, users logged into each host, and events capturing commands executing files on each of the hosts. The plan to identify other hosts impacted by this network compromise is the build a network graph to show relationships between network hosts and commands executed on the network using a search query to traverse the graph.

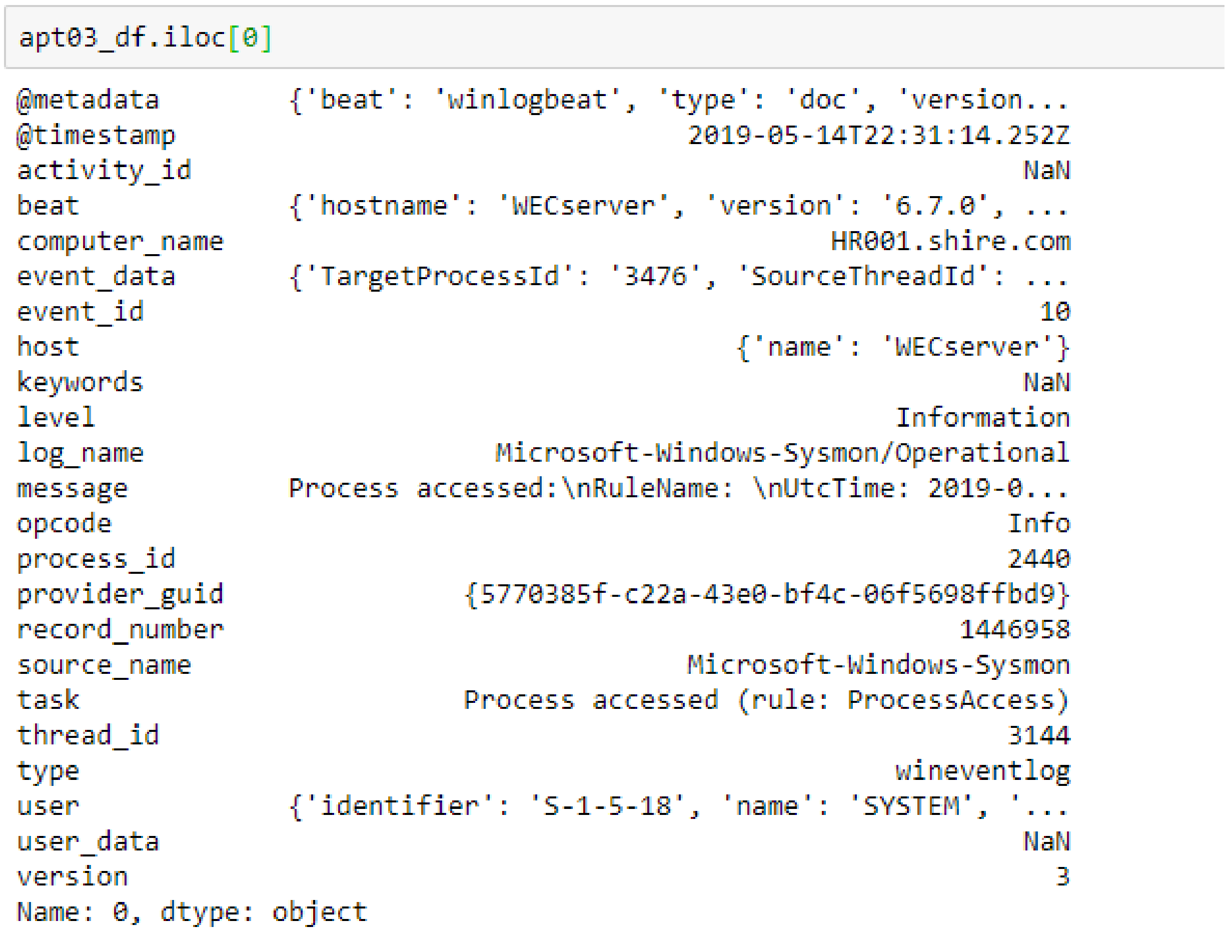
Data Understanding

Dimensions of the dataset shown in Figure 13 with 23 different attributes with nested entries and 121659 instances. Figure 14 shows a printout of the first record demonstrates some of the attributes with nested entries.



```
apt03_df.shape
(121659, 23)
```

Figure 13 APT03 Dataset Dimensions

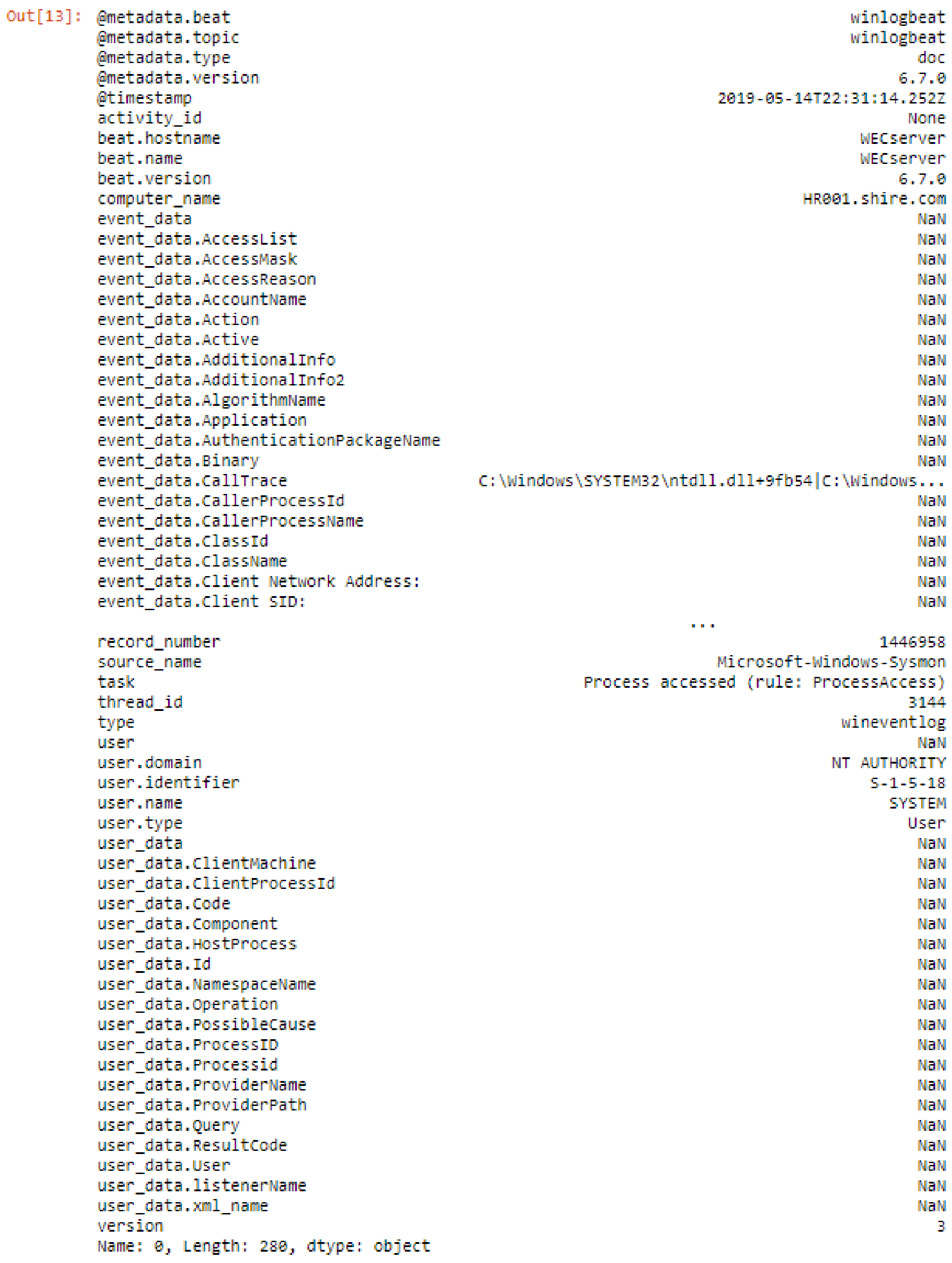


```
apt03_df.iloc[0]
```

```
@metadata         {'beat': 'winlogbeat', 'type': 'doc', 'version...
@timestamp                                2019-05-14T22:31:14.252Z
activity_id                                                    NaN
beat              {'hostname': 'WECserver', 'version': '6.7.0', ...
computer_name                                      HR001.shire.com
event_data        {'TargetProcessId': '3476', 'SourceThreadId': ...
event_id                                                        10
host                                         {'name': 'WECserver'}
keywords                                                       NaN
level                                                  Information
log_name                      Microsoft-Windows-Sysmon/Operational
message           Process accessed:\nRuleName: \nUtcTime: 2019-0...
opcode                                                        Info
process_id                                                    2440
provider_guid               {5770385f-c22a-43e0-bf4c-06f5698ffbd9}
record_number                                              1446958
source_name                                Microsoft-Windows-Sysmon
task                          Process accessed (rule: ProcessAccess)
thread_id                                                     3144
type                                                   wineventlog
user              {'identifier': 'S-1-5-18', 'name': 'SYSTEM', '...
user_data                                                      NaN
version                                                          3
Name: 0, dtype: object
```

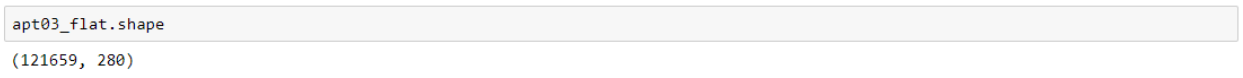
Figure 14 APT03 with Nested Attributes

In order to properly model the relationships between different network events the nested attributes are flattened. Figure 15 shows an example of a flattened attribute. In the process of flattening the shape of the dataset is now 280 attributes with the expected same number of 121,659 instances shown in Figure 16.



```
Out[13]: @metadata.beat                                                          winlogbeat
         @metadata.topic                                                         winlogbeat
         @metadata.type                                                                 doc
         @metadata.version                                                            6.7.0
         @timestamp                                                2019-05-14T22:31:14.252Z
         activity_id                                                                   None
         beat.hostname                                                            WECserver
         beat.name                                                                WECserver
         beat.version                                                                 6.7.0
         computer_name                                                        HR001.shire.com
         event_data                                                                     NaN
         event_data.AccessList                                                          NaN
         event_data.AccessMask                                                          NaN
         event_data.AccessReason                                                        NaN
         event_data.AccountName                                                         NaN
         event_data.Action                                                              NaN
         event_data.Active                                                              NaN
         event_data.AdditionalInfo                                                      NaN
         event_data.AdditionalInfo2                                                     NaN
         event_data.AlgorithmName                                                       NaN
         event_data.Application                                                         NaN
         event_data.AuthenticationPackageName                                           NaN
         event_data.Binary                                                              NaN
         event_data.CallTrace                       C:\Windows\SYSTEM32\ntdll.dll+9fb54|C:\Windows...
         event_data.CallerProcessId                                                     NaN
         event_data.CallerProcessName                                                   NaN
         event_data.ClassId                                                             NaN
         event_data.ClassName                                                           NaN
         event_data.Client Network Address:                                             NaN
         event_data.Client SID:                                                         NaN
                                                                    ...
         record_number                                                              1446958
         source_name                                                   Microsoft-Windows-Sysmon
         task                                           Process accessed (rule: ProcessAccess)
         thread_id                                                                     3144
         type                                                                   wineventlog
         user                                                                           NaN
         user.domain                                                           NT AUTHORITY
         user.identifier                                                           S-1-5-18
         user.name                                                                   SYSTEM
         user.type                                                                     User
         user_data                                                                      NaN
         user_data.ClientMachine                                                        NaN
         user_data.ClientProcessId                                                      NaN
         user_data.Code                                                                 NaN
         user_data.Component                                                            NaN
         user_data.HostProcess                                                          NaN
         user_data.Id                                                                   NaN
         user_data.NamespaceName                                                        NaN
         user_data.Operation                                                            NaN
         user_data.PossibleCause                                                        NaN
         user_data.ProcessID                                                            NaN
         user_data.Processid                                                            NaN
         user_data.ProviderName                                                         NaN
         user_data.ProviderPath                                                         NaN
         user_data.Query                                                                NaN
         user_data.ResultCode                                                           NaN
         user_data.User                                                                 NaN
         user_data.listenerName                                                         NaN
         user_data.xml_name                                                             NaN
         version                                                                          3
         Name: 0, Length: 280, dtype: object
```

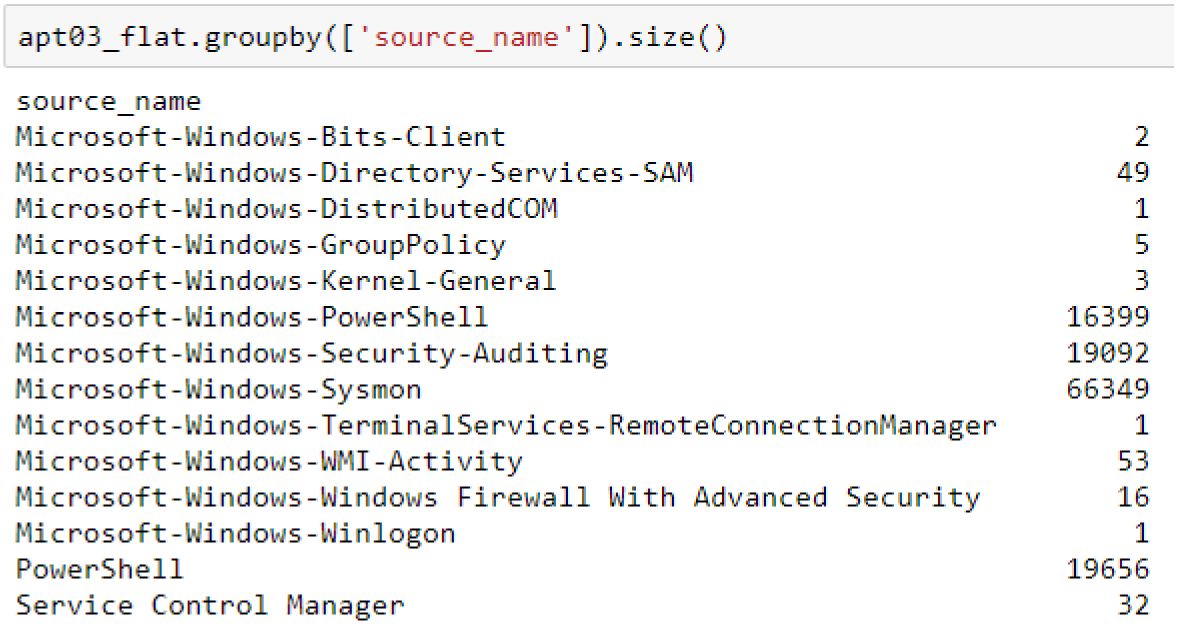
Figure 15 Flattened Attribute



```
apt03_flat.shape
(121659, 280)
```

Figure 16 Dimensions with Flattened Attributes

Now the sources of network logs can be found in the “source_name” attribute. Figure 17 shows the different log sources and record counts. Note there are two source names for PowerShell events.



```
apt03_flat.groupby(['source_name']).size()
```

```
source_name
Microsoft-Windows-Bits-Client                                       2
Microsoft-Windows-Directory-Services-SAM                           49
Microsoft-Windows-DistributedCOM                                    1
Microsoft-Windows-GroupPolicy                                       5
Microsoft-Windows-Kernel-General                                    3
Microsoft-Windows-PowerShell                                    16399
Microsoft-Windows-Security-Auditing                             19092
Microsoft-Windows-Sysmon                                        66349
Microsoft-Windows-TerminalServices-RemoteConnectionManager          1
Microsoft-Windows-WMI-Activity                                     53
Microsoft-Windows-Windows Firewall With Advanced Security          16
Microsoft-Windows-Winlogon                                          1
PowerShell                                                      19656
Service Control Manager                                            32
```

Figure 17 Log Sources and Record Counts

To ease the data analysis, we write the data frame out to a .CSV file, then split into thirteen (13) separate files grouped by source_name. The original file was inclusive to every log source there were attributes explicit to specific source types. After splitting into separate files split by source_name unused attributes were removed to better understand what the information captured in each data attribute. Table 8 Attributes per Log Source shows the list of attributes for each source_name.

For this step the dataset is evaluated to determine relevant information. This step the fourteen log sources are separated into thirteen log files, combining Microsoft PowerShell and PowerShell events into a single file. From here the relevant attributes are identified for hosts logged onto the network shown in Table 1, username account logging from Windows Security Auditing, and process events captured in the logged event's event data.

Table 1 Network Graph Hosts

computer_name	IPAddress	Platform	Version	Purpose
HFDC01.shire.com	172.18.39.5	Windows	Win 2019	DC
HR001.shire.com	172.18.39.106	Windows	Win 10	Client
IT001.shire.com	172.18.39.105	Windows	Win 10	Client
ACCT001.shire.com	172.18.39.100	Windows	Win 10	Client
WECServer.shire.com	172.18.39.102	Windows	Win 2019	Log Collector
FILE001.shire.com	172.18.39.103	Windows	Win 2019	File Server
HELK	172.18.39.6	Linux	Ubuntu 18	Data Analysis
RTO	172.18.39.8	Linux	Ubuntu 18	Red Team C2

Data Preparation

To build the network graph model the relevant data is extracted from the data logs. Hosts are defined as categorical. Command lines are treated as strings to accommodate the data structure of folders and data paths as free text. The next step would be to dissect the command line into two attributes: one for data path and second for command, for this exercise the actual algorithm is a search the attribute is not split. The result is three data files: 1) hosts on the network from Table 1, 2) user account log in activity, and 3) command and process activity initiated in Microsoft Windows Security Auditing, Microsoft Windows Sysmon, and PowerShell log sources.

Data Modeling

The intention for this graph model is to search for a command then return the host that has the relationship of "generate" the command event. First generate the separate groups of nodes then create relationships linking the host node to each event it generated within the datasets.

Referring to Figure 7 Example Information Domains begin by generating nodes for the hosts. Using Table 1 Network Graph Hosts a group of nodes called Host is generated with a property tag for each hostname shown in Table 2. For larger networks tags should be added to further document the profile of each node.

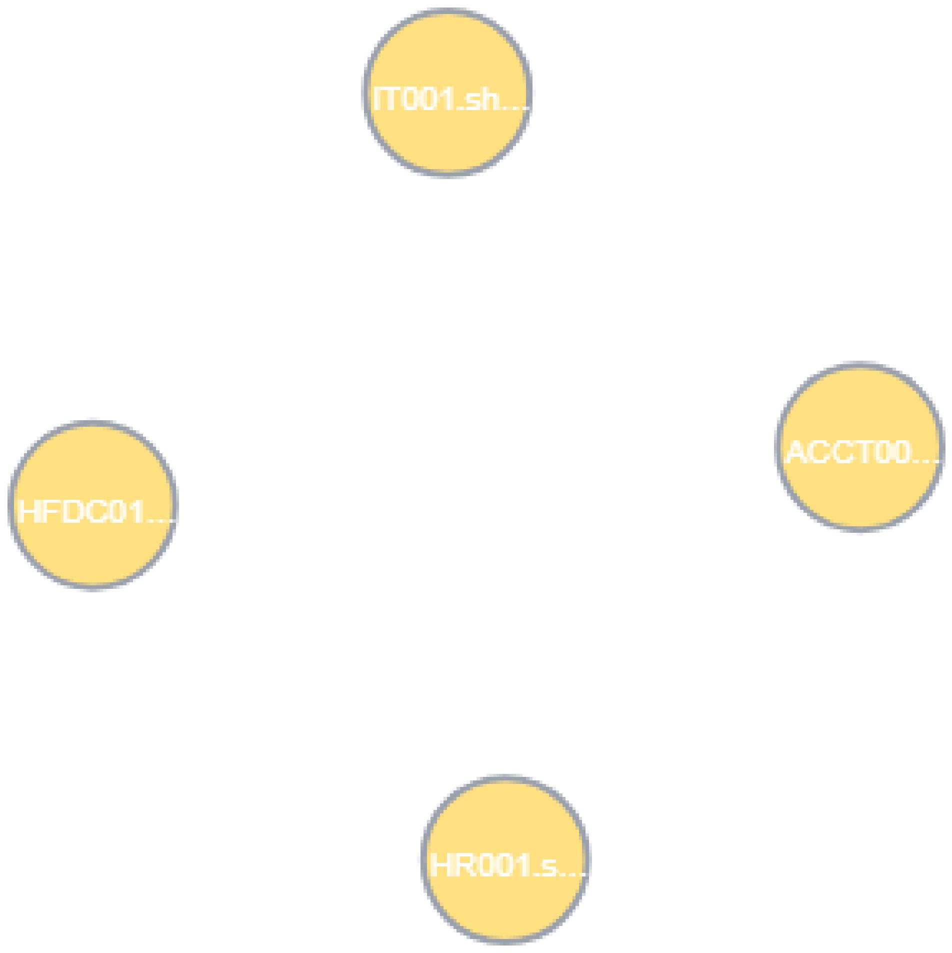
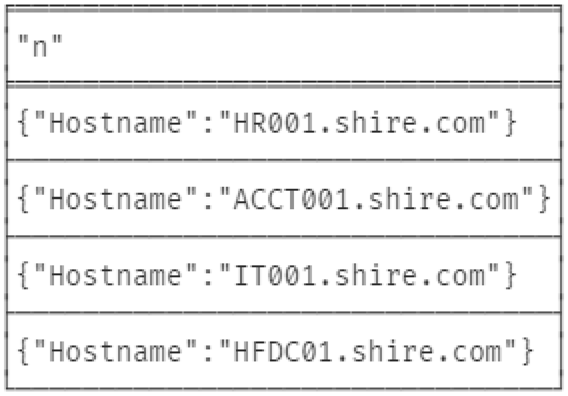


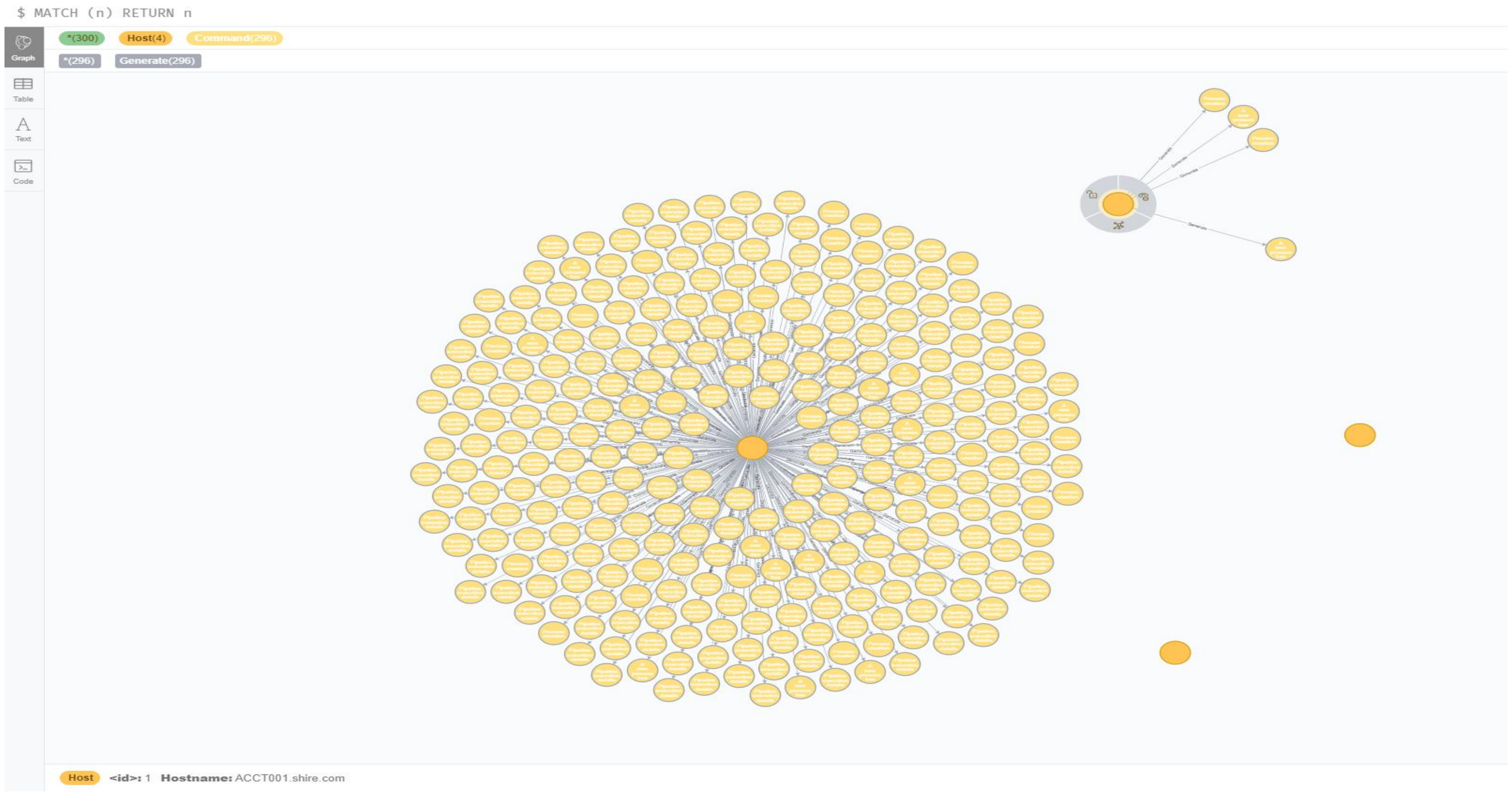
Figure 18 Host Nodes

Table 2 Generated Host Nodes



"n"
{"Hostname":"HR001.shire.com"}
{"Hostname":"ACCT001.shire.com"}
{"Hostname":"IT001.shire.com"}
{"Hostname":"HFDC01.shire.com"}

Next the group of nodes Commands with 18,748 unique nodes generated. A directed relationship of “Generate” connects the source hostname to every command generated. This logic results in a one to many relationships between hosts to generated commands but an unto relationship for every command node as shown in Figure 19 Partial Generated Network Graph, noting due to the high number of commands only the first 300 results are returned.


```
$ MATCH (n) RETURN n
```

Host <id>: 1 Hostname: ACCT001.shire.com

Figure 19 Partial Generated Network Graph

Evaluation

Using Cypher queries to evaluate this network graph model, we perform a search query to first find what Event Commands contain the file name listed as an IoC of "autoupdate.vbs". The Cypher query to query the graph then return the Event commands is below, with the results

```
MATCH (h)-[:Generate]->(n) WHERE n.command CONTAINS "autoupdate.vbs" RETURN n
```

Table 3 Network Graph Return Command Events

n (Returned Command)
{command:C:\Windows\System32\WScript.exe C:\Users\nmartha\Downloads\autoupdate.vbs ,eventname:A new process has been created}
{eventname:Process creation,command:C:\Windows\System32\WScript.exe C:\Users\nmartha\Downloads\autoupdate.vbs }
{eventname:Process creation,command:cmd.exe /c C:\windows\system32\autoupdate.vbs}
{command:cmd.exe /c C:\windows\system32\autoupdate.vbs,eventname:A new process has been created}
{command:C:\Windows\System32\WScript.exe C:\windows\system32\autoupdate.vbs ,eventname:A new process has been created}
{eventname:Process creation,command:C:\Windows\System32\WScript.exe C:\windows\system32\autoupdate.vbs }
{eventname:Pipeline execution details for command line: Write-Host Test.,command:move-item c:\windows\system32\autoupdate.vbs FileSystem:\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs}
{eventname:Pipeline execution details for command line: Write-Host Test.,command:COPY c:\Windows\System32\autoupdate.vbs `\`\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs}
{command:C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto,eventname:A new process has been created}
{eventname:Process creation,command:C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater Table 3. (continued) [illegible] **C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto}**
{eventname:Pipeline execution details for command line: Write-Host Test.,command:type FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs}
{eventname:Process creation,command:cmd.exe /c C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs }
{eventname:Process creation,command:C:\Windows\System32\WScript.exe C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs }

With a slight modification to the previous Cypher query to further evaluate this network graph model, we perform a search query to first identify what Event Commands contain the file name listed as an IoC of "autoupdate.vbs". Th Cypher query to query the graph returns the hostnames generating the earlier command events

```
MATCH (h)-[:Generate]->(n) WHERE n.command CONTAINS "autoupdate.vbs" RETURN h
```

Table 4 Network Graph Return Hostnames

H (Returned Hosts)
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HR001.shire.com}
{hostname:HFDC01.shire.com}
{hostname:HFDC01.shire.com}

Results Summary

Walking through this exercise we defined a question to research using an identified dataset based on a documented APT03 TTP. The forensic evidence described a filename of interest of “autoupdate.vbs” with an attack vehicle of PowerShell. Combining this knowledge, we examine PowerShell logs for occurrences of commands acted on filenames titled “autoupdate.vbs”. Given this prior knowledge, relevant data was found and ingested into a graph database to generate a network graph model with a relationship mapping which hosts generated the commands. Finally, queries were generated to search the graph database returning all events related to the filename of interest then return the hostnames finding other hosts affected by the initial incident response investigation.

Usecase #2: Lateral Movement

The results from Usecase #1 is a list of PowerShell commands acted on a filename titled "autoupdate.vbs". In this usecase we further evaluate the data using the same network graph but dive further into the commands and how to map out the steps of a network attack. Since this usecase utilizes the same dataset the Business Understanding, Data Understanding, and Data Preparation steps are repeated from Usecase #1.

Mapping back to the MITRE ATT@CK Framework the attack steps are elements within the Lateral Movement stage of an attack using Command and Control, Persistence, and Execution of an attack vector. Table 3 Network Graph Return Command Events documents analysis results to find commands of interest pertaining to the initial filename of interest. Combining this list plus prior knowledge that the attack vehicle is Powershell the sequence of commands begins to outline the behaviors within the network attack.

Beginning with verifying the role each host from Table 4 Network Graph Return Hostnames the query targets identifying hosts that generated three different commands defined as IoC including one of a file or information transfer to a second host, then confirming activity on the second host generated a command defined as an IoC. Table 5 Returned Results Generated Commands the results returned by the following query. The query then creates a relationship from the first host to the second host called Pivot, indicating lateral movement of the malicious activity.

- The file "autoupdate.vbs" is copied to a remote location using the Windows PowerShell commandlet move-item.
- A new service titled "Adobe Flash Updater" is created using the Windows sc.exe command on the "autoupate.vbs" file.
- Finally, the Adobe Flash Updater service is executed.

```
	MATCH (event1 {command: "move-item c:\\windows\\system32\\autoupdate.vbs
FileSystem::\\\\HFDC01\\C$\\Users\\pgustavo\\AppData\\Roaming\\Adobe\\Flash
Player\\autoupdate.vbs"})<-[: Generate]-(host)-[: Generate]->(event2 {command:
"C:\\WINDOWS\\system32\\sc.exe \\\\HFDC01 create AdobeUpdater binPath= cmd.exe /c
\\C:\\Users\\pgustavo\\AppData\\Roaming\\Adobe\\Flash Player\\autoupdate.vbs\\
DisplayName= Adobe Flash Updater start= auto"})
	MATCH (host)-[:Generate]->(event3 {command: "C:\\Windows\\system32\\sc.exe
\\\\HFDC01 start AdobeUpdater"})
	RETURN host, event1, event2, event3
```

Table 5 Returned Results Generated Commands

host	event1	event2	event3
{hostname:HR001.shire.com}	{eventname:Pipeline execution details for command line: Write-Host Test. command:move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs}	{eventname:Process creation command:C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto}	{eventname:Process creation command:C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater}
{hostname:HR001.shire.com}	{eventname:Pipeline execution details for command line: Write-Host Test. command:move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs}	{eventname:Process creation command:C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto}	{eventname:Process creation command:C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater}
{hostname:HR001.shire.com}	{eventname:Pipeline execution details for command line: Write-Host Test. command:move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs}	{eventname:Process creation command:C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto}	{eventname:Process creation command:C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater}

Table 5. (continued)

host	**event1**	**event2**	**event3**
{hostname:HR001.shire.com}	{eventname:Pipeline execution details for command line: Write-Host Test. command:move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs}	{eventname:Process creation command:C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto}	{eventname:Process creation command:C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater}

Figure 20 Returned Results Generated Commands shows the subgraph of the query results showing HR001 documenting the relationship of generating events documenting the commands for copying the malicious file and creating a service by executing the malicious file.

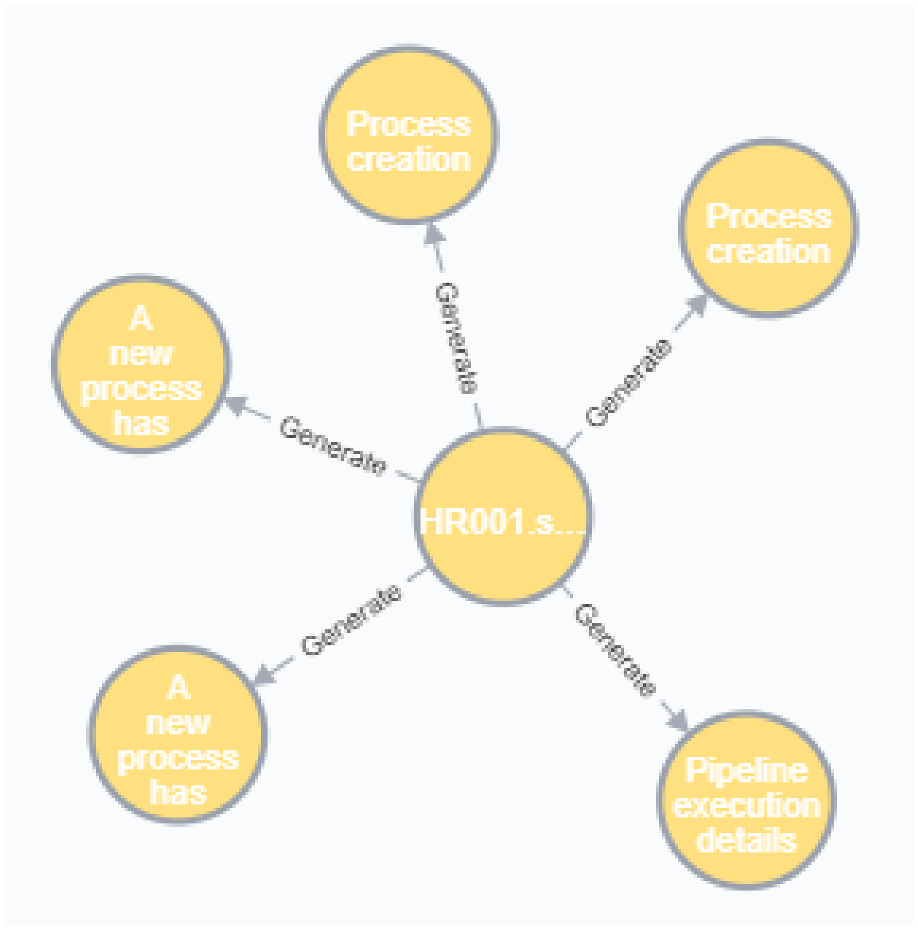


Figure 20 Returned Results Generated Commands

Lateral Movement Assessment

Examining the patterns in the commands there are two hosts of interest, showing at some point the malicious file was executed on another host indicating a potential pivot. This pattern is more complex, investigating between activity starting on one host the confirming similar actions on a another. To explore this, we build onto the earlier Cypher query by adding an additional MATCH parameter searching for the "sc.exe" command to start the same service. This query generates a new relationship between the source host identified in Figure 20 and target host matching this additional parameter called "Pivot".

```
MATCH (event1 {command: "move-item c:\\windows\\system32\\autoupdate.vbs FileSystem::\\\\HFDC01\\C$\\Users\\pgustavo\\AppData\\Roaming\\Adobe\\Flash Player\\autoupdate.vbs"})<-[: Generate]-(host)-[: Generate]->(event2 {command: "C:\\WINDOWS\\system32\\sc.exe \\\\HFDC01 create AdobeUpdater binPath= cmd.exe /c \\C:\\Users\\pgustavo\\AppData\\Roaming\\Adobe\\Flash Player\\autoupdate.vbs\\ DisplayName= Adobe Flash Updater start= auto"})
MATCH (host)-[:Generate]->(event3 {command: "C:\\Windows\\system32\\sc.exe \\\\HFDC01 start AdobeUpdater"})
MATCH (t)-[:Generate]->(n)
WHERE n.command="cmd.exe /c C:\\Users\\pgustavo\\AppData\\Roaming\\Adobe\\Flash Player\\autoupdate.vbs "
CREATE (host)-[r:Pivot]->(t)
RETURN host, event1.command, event2.command, event3.command, t, n.command
```

Table 6 documents the results of this query, where the host generating the event1.command, event2.command, and event3.command. The t is found target generating the n.command event.

Table 6 Pivot Relationship Hosts

host	event1.command	event2.command	event3.command	t	n.command
{hostname:HR001. shire.com}	move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs	C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto	C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater	{hostname:HFDC01.shire.com}	cmd.exe /c C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs
{hostname:HR001. shire.com}	move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs	C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto	C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater	{hostname:HFDC01.shire.com}	cmd.exe /c C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs
{hostname:HR001. Table 6. (continued)	move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs	C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto	C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater	{hostname:HFDC01.shire.com}	cmd.exe /c C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs
{hostname:HR001. shire.com}	move-item c:\windows\system32\autoupdate.vbs FileSystem::\\HFDC01\C$\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs	C:\WINDOWS\system32\sc.exe \\HFDC01 create AdobeUpdater binPath= cmd.exe /c \C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs\ DisplayName= Adobe Flash Updater start= auto	C:\Windows\system32\sc.exe \\HFDC01 start AdobeUpdater	{hostname:HFDC01.shire.com}	cmd.exe /c C:\Users\pgustavo\AppData\Roaming\Adobe\Flash Player\autoupdate.vbs

Figure 21 shows the subgraph of the query results showing HR001 generating the command to copy and move the “autoupdate.vbs” file to HFDC01, the a command on HFDCO01 to execute the same service, hence the pivot from HF001.shire.com to HFDC01.shire.com.This further supports the findings in Results Summary.

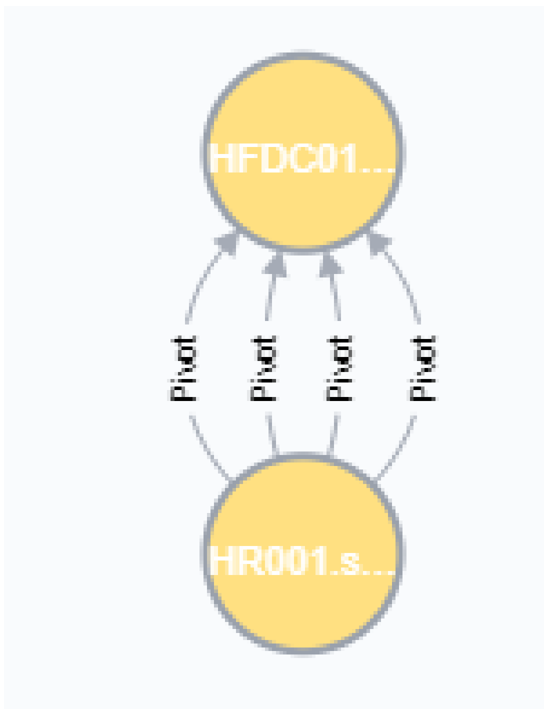


Figure 21 Pivot Relationship Hosts

Results Summary

This use case expands on the examination results from Use case #1: Filename Indicator of Compromise, using Cypher to generate an attack query. The attack query discovers more details on the anatomy of the network attack by exploring actual executed commands and actions to provide a more detailed modeling of a network attack. Finally, we build a relationship based on the executed command and actions to find pivoting of the malicious behavior finding potential lateral movement.

Anatomy of a Cypher Query

This section breaks down an example Cypher query with commentary to enable understanding the anatomy of the more complex Cypher query used in Use case #2. This query targets identifying hosts that generated three different commands defined as IoC including one of a file or information transfer to a second host, then confirming activity on the second host generated a command defined as an IoC..The query then creates a relationship from the first host to the second host called Pivot, indicating lateral movement of the malicious activity.

Table 7 Cyber Query Anatomy

Description	**Cypher Query**
MATCH query searches for any host which executed two of the identified commands that make up this attack chain's IOC. Commands are related to hosts using the "Generate" relationship.	• MATCH (event1 {command: "move-item c:\\windows\\system32\\autoupdate.vbs FileSystem::\\\\HFDC01\\C$\\Users\\pgustavo\\AppData\\Roaming\\Adobe\\Flash Player\\autoupdate.vbs"})<-[: Generate]-(host)-[: Generate]->(event2 {command: "C:\\WINDOWS\\system32\\sc.exe \\\\HFDC01 create AdobeUpdater binPath= cmd.exe /c \\C:\\Users\\pgustavo\\AppData\\Roaming\\Adobe\\Flash Player\\autoupdate.vbs\\ DisplayName= Adobe Flash Updater start= auto"})
The second MATCH query further refines the prior set of hosts by limiting the result to only those who also executed the third command in the IOC.	• MATCH (host)-[:Generate]->(event3 {command: "C:\\Windows\\system32\\sc.exe \\\\HFDC01 start AdobeUpdater"})

Table 7 (continued)

Description	**Cypher Query**
The second MATCH query further refines the prior set of hosts by limiting the result to only those who also executed the third command in the IOC.	• MATCH (target)-[:Generate]->(n) WHERE n.command="cmd.exe /c C:\\\\Users\\\\pgustavo\\\\AppData\\\\Roaming\\\\Adobe\\\\Flash Player\\\\autoupdate.vbs"
The inputs are the list of hosts that with confirmed compromise IoC and targets that generated events showing execution of the malicious file. **CREATE generates a relationship from source hosts to target hosts.**	• CREATE (host)-[r:Pivot]->(t)

CHAPTER 6. RESEARCH CONTRIBUTIONS

This dissertation provides a framework for incident response investigators to more fully contain and eradicate a detected network compromise by using graph databases to efficiently query network and host-based traffic stored in hot or warm storage. Further by walking through the steps documented in *Figure 1 Procedure to Identify Impacted Hosts* two gaps were identified:

1. Finding how to, given a specific attack query, prepare data taken from storage to build a network graph model.
2. Prior to constructing a network graph taking multisensory data and fuse into a format given the data in storage is a mix of formats, structures, and even unstructured data.

This dissertation provides a framework of how to identify relevant and applicable data sources then fusing the data into a format that can be ingested to generate a network graph model modeling events to impacted hosts or accounts, where appropriate. Finally, the structure of the network graph is the first step towards enabling application of added advanced analytics or graph learning.

CHAPTER 7. FUTURE WORK

Graph databases offer an efficient method to model network and host-based traffic, accommodating the interrelations of the many data sources. The logical next steps for this work include:

- Integrating temporal attributes into the network graph model enabling better behavioral based analysis based on sequence of events per host.
- Considering this is the initial step to generating a graph-based network model, reuse the approach to integrate with knowledge graphs allowing modeling of not only network hosts and events, but also procedure flows such as identify and access management or vulnerability management.
- Along with knowledge graphs, explore application of graph learning to the generated network graph model. This takes into consideration analysis of information systems goes beyond straight data science but delves into network science. Essentially, monitoring and assessing how data flows over time across different physical locations which begins to resemble network science.

CHAPTER 8. GENERAL CONCLUSION

Current network monitoring technologies do not keep up with increasing size and complexity of log data being monitored due to the ever-quickening evolution of adversary tactics. Network monitoring architectures and tactics must adapt to accommodate the increasing complexities and volumes of network data. To accommodate this Computer Network Defense teams use analytics for network monitoring activities such as cyber threat hunting. Challenges of this adaptation results in increased volumes and types of network and host-based security data leaving CND teams dealing with data mining and data modeling challenges. This dissertation presents a framework to fuse data from different source types using graph databases. Two use cases present a walkthrough of applying the framework and presents how to use queries taking advantage of the efficiencies provided by graph databases.

Table 8 Attributes per Log Source

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
activity_id		X	X	X	X	X				X			X
beat.hostname	X	X	X	X	X								
beat.name	X	X	X	X	X								
beat.version	X	X	X	X	X								
computer_name	X	X	X	X	X	X	X	X	X	X	X	X	X
host.name	X	X	X	X	X	X	X	X	X	X	X	X	X
ID	X	X	X	X	X	X	X	X	X	X	X	X	X
keywords	X	X				X						X	X
level	X	X	X	X	X	X	X	X	X	X	X	X	X
log_name	X	X	X	X	X	X	X	X	X	X	X	X	X
message	X	X	X	X	X	X	X	X	X	X	X	X	X
metadata.beat	X	X	X	X	X								
metadata.topic	X	X	X	X	X								
metadata.type	X	X	X	X	X								
metadata.version	X	X	X	X	X								
opcode	X	X	X	X	X	X	X	X	X	X	X	X	X
process_id	X	X	X	X	X	X	X	X	X	X	X	X	X
provider_guid	X	X	X	X	X	X	X	X	X	X	X	X	X
record_number	X	X	X	X	X	X	X	X	X	X	X	X	X
source_name	X	X	X	X	X	X	X	X	X	X	X	X	X
task		X				X			X				X
thread_id	X	X	X	X	X	X	X	X	X	X	X	X	X
Timestamp	X	X		X	X	X	X	X	X	X	X	X	X
type	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data													
event_data.AccessList													
event_data.AccessMask						X							
event_data.AccessReason						X							
event_data.AccountName												X	
event_data.Action								X					
event_data.Active								X					
event_data.AdditionalInfo					X								
event_data.AdditionalInfo2					X								
event_data.AlgorithmName													
event_data.Application													
event_data.AuthenticationPackageName													
event_data.Binary												X	
event_data.CallerProcessId					X								
event_data.CallerProcessName					X								
event_data.CallTrace									X				
event_data.ClassId													
event_data.ClassName													
event_data.Client Network Address:					X								
event_data.Client SID:					X								
event_data.ClientProcessId													
event_data.CommandLine		X						X					
event_data.Company								X					
event_data.CompatibleIds													

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.ContextInfo		X											
event_data.CountOfCredentialsReturned													
event_data.CreationUtcTime								X					
event_data.CurrentDirectory								X					
event_data.DCName										X			
event_data.Description								X					
event_data.DestAddress													
event_data.DestinationHostname								X					
event_data.DestinationIp									X				
event_data.DestinationIsIpv6								X					
event_data.DestinationPort									X				
event_data.DestinationPortName								X					
event_data.DestPort													
event_data.Details								X					
event_data.Device								X					
event_data.DeviceDescription													
event_data.DeviceId													
event_data.Direction								X					
event_data.DirtyPages							X						
event_data.DisabledPrivilegeList													
event_data.EdgeTraversal								X					
event_data.ElevatedToken													

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.EmbeddedContext								X					
event_data.EnabledPrivilegeList													
event_data.ErrorCode										X			
event_data.ErrorDescription										X			
event_data.EventCountTotal													
event_data.EventIdx													
event_data.EventType								X					
event_data.ExtensionId										X			
event_data.ExtensionName										X			
event_data.FailureReason													
event_data.FileVersion								X					
event_data.FilterRTID													
event_data.Flags								X					
event_data.GrantedAccess									X				
event_data.GroupMembership													
event_data.HandleId						X							
event_data.Hash								X					
event_data.Hashes								X					
event_data.HiveName							X						
event_data.HiveNameLength							X						
event_data.Image								X					
event_data.ImageLoaded								X					
event_data.ImagePath												X	

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.ImpersonationLevel													
event_data.Initiated								X					
event_data.IntegrityLevel								X					
event_data.IpAddress													
event_data.IpPort													
event_data.KeyFilePath													
event_data.KeyLength													
event_data.KeyName													
event_data.KeysUpdated							X						
event_data.KeyType													
event_data.LayerName													
event_data.LayerRTID													
event_data.LmPackageName													
event_data.LocalAddresses								X					
event_data.LocalOnlyMapped								X					
event_data.LocationInformation													
event_data.LogonGuid								X					
event_data.LogonId								X					
event_data.LogonProcessName													
event_data.LogonType													
event_data.LooseSourceMapped								X					
event_data.MandatoryLabel													

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.MessageNumber		X											
event_data.MessageTotal		X											
event_data.ModifyingApplication								X					
event_data.ModifyingUser								X					
event_data.NewProcessId													
event_data.NewProcessName													
event_data.NewSd													
event_data.NewState													
event_data.number				X									
event_data.ObjectName													
event_data.ObjectServer						X							
event_data.ObjectType													
event_data.OldSd													
event_data.Operation													
event_data.OperationType													
event_data.Origin													
event_data.PackageName													
event_data.param1	X	X										X	
event_data.param10	X												
event_data.param11	X												
event_data.param2	X	X										X	
event_data.param3	X	X										X	
event_data.param4	X											X	
event_data.param5	X												
event_data.param6	X												

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.param7	X												
event_data.param8	X												
event_data.param9	X												
event_data.ParentCommandLine								X					
event_data.ParentImage								X					
event_data.ParentProcessGuid								X					
event_data.ParentProcessId								X					
event_data.ParentProcessName													
event_data.Path		X											
event_data.Payload		X											
event_data.PipeName								X					
event_data.PreAuthType													
event_data.PreviousCreationUtcTime								X					
event_data.PrivilegeList						X							
event_data.ProcessCreationTime													
event_data.ProcessGuid									X				
event_data.ProcessId						X		X					
event_data.ProcessID						X		X					
event_data.ProcessingMode										X			
event_data.ProcessingTimeInMilliseconds										X			
event_data.ProcessName						X							
event_data.Product								X					

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.ProfileChanged													
event_data.Profiles								X					
event_data.Properties													
event_data.Protocol						X		X	X				
event_data.ProviderName													
event_data.ReadOperation													
event_data.RelativeTargetName													
event_data.RemoteAddresses								X					
event_data.RemoteMachineID													
event_data.RemoteUserID													
event_data.ResourceAttributes						X							
event_data.ResourceManager													
event_data.RestrictedAdminMode													
event_data.RestrictedSidCount						X							
event_data.ReturnCode													
event_data.RuleId								X					
event_data.RuleName								X					
event_data.RuleStatus								X					
event_data.RunspaceId		X											
event_data.SamAccountName													

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.SchemaVersion								X					
event_data.ScriptBlockId		X											
event_data.ScriptBlockText		X											
event_data.SecurityOptions								X					
event_data.Service													
event_data.ServiceAccount													
event_data.ServiceFileName													
event_data.ServiceName												X	
event_data.ServiceSid													
event_data.ServiceStartType													
event_data.ServiceType												X	
event_data.SessionId													
event_data.ShareLocalPath													
event_data.ShareName													
event_data.SidHistory													
event_data.Signature								X					
event_data.SignatureStatus								X					
event_data.Signed								X					
event_data.SourceAddress													
event_data.SourceHandleId		X				X							
event_data.SourceHostname									X				
event_data.SourceImage									X				
event_data.SourceIp									X				
event_data.SourceIsIpv6									X				
event_data.SourcePort						X		X					
event_data.SourcePortName									X				

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.SourceProcessGUID									X				
event_data.SourceProcessId						X			X				
event_data.SourceThreadId		X							X				X
event_data.StartType												X	
event_data.Status													
event_data.SubjectDomainName						X							
event_data.SubjectLogonId						X							
event_data.SubjectUserName						X							
event_data.SubjectUserSid						X							
event_data.SubStatus													
event_data.SupportInfo1										X			
event_data.SupportInfo2										X			
event_data.TargetDomainName													
event_data.TargetFilename								X					
event_data.TargetHandleId						X							
event_data.TargetImage									X				
event_data.TargetInfo													
event_data.TargetLinkedLogonId													
event_data.TargetLogonGuid													
event_data.TargetLogonId													
event_data.TargetName													
event_data.TargetObject								X					

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.TargetOutboundDomainName													
event_data.TargetOutboundUserName													
event_data.TargetProcessGUID									X				
event_data.TargetProcessId		X		X		X	X		X	X			X
event_data.TargetServerName													
event_data.TargetSid													
event_data.TargetUserName													
event_data.TargetUserSid													
event_data.TaskContentNew													
event_data.TaskName													
event_data.TerminalSessionId								X					
event_data.TicketEncryptionType													
event_data.TicketOptions													
event_data.TokenElevationType													
event_data.TransactionId						X							
event_data.TransmittedServices													
event_data.TSId											X		
event_data.Type													
event_data.User								X					

Table 8 (continued)

Attributes\Log Sources	Distributed COM	PowerShell	TerminalServices-Remote Connection Manager	Bits-Client	Directory-Services-SAM	Security-Auditing	Kernel-General	Windows Firewall	Sysmon	Group Policy	Win logon	Service Control Manager	WMI-Activity
event_data.UserSid											X		
event_data.UtcTime									X				
event_data.VendorIds													
event_data.VirtualAccount													
event_data.Workstation													
event_data.WorkstationName													
user													
user.domain	X	X	X	X	X		X	X	X	X	X	X	X
user.identifier	X	X	X	X	X		X	X	X	X	X	X	X
user.name	X	X	X	X	X		X	X	X	X	X	X	X
user.type	X	X	X	X	X		X	X	X	X	X	X	X
user_data		X		X									
user_data.ClientMachine													X
user_data.ClientProcessId													X
user_data.Code													X
user_data.Component													X
user_data.HostProcess													X
user_data.Id													X
user_data.listenerName			X										
user_data.NamespaceName													
user_data.Operation													X
user_data.PossibleCause													X
user_data.ProcessID													X
user_data.Processid													X
user_data.ProviderName													X
user_data.ProviderPath													X
user_data.Query													

Table 8 (continued)

Attributes\Log Sources	**Distributed COM**	**PowerShell**	**TerminalServices-Remote Connection Manager**	**Bits-Client**	**Directory-Services-SAM**	**Security-Auditing**	**Kernel-General**	**Windows Firewall**	**Sysmon**	**Group Policy**	**Win logon**	**Service Control Manager**	**WMI-Activity**
user_data.ResultCode													X
user_data.User													
user_data.xml_name			X										X
version	X	X	X	X	X	X	X	X	X		X	X	X

Table 8. (continued)

CHAPTER 10. REFERENCES

[1] Gartner, "Gartner Glossary," Gartner, 2019. [Online]. Available: https://www.gartner.com/en/information-technology/glossary/security-orchestration-automation-response-soar. [Accessed 2019].

[2] D. Cearley and B. Burke, "Top 10 Strategic Technology Trends for 2019," Gartner, 2018.

[3] F. Tong, "Towards Data Science," 23 April 2019. [Online]. Available: https://towardsdatascience.com/graph-theory-and-deep-learning-know-hows-6556b0e9891b. [Accessed 7 August 2019].

[4] Kelley, Diana and John, Siân, "Microsoft Security," Microsoft, 30 August 2018. [Online]. Available: https://www.microsoft.com/security/blog/2018/08/30/building-the-security-operations-center-of-tomorrow-harnessing-the-law-of-data-gravity/. [Accessed 08 2019].

[5] The MITRE Corporation, "Finding Cyber Threats with ATT@CK Based Analytics," June 2017. [Online]. Available: https://www.mitre.org/sites/default/files/publications/16-3713-finding-cyber-threats%20with%20att%26ck-based-analytics.pdf. [Accessed 2018].

[6] e. a. Manos Antonakakis, "Understanding the Mirai Botnet," in *26th USENIX Security Symposium*, Van Courver, BC, 2017.

[7] R. S. a. V. Paxsom, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *IEEE Symposium on Security and Privacy*, 2010.

[8] M. Rosenquis, "Cyber security Hunter teams are the next advancement in network defense," Intel, 28 November 2012. [Online]. Available: https://itpeernetwork.intel.com/cyber-security-hunter-teams-are-the-next-advancement-i n-network-defense/#gs.2GHfyV8M. . [Accessed 2019].

[9] S. Chang, " Securing enterprise networks with statistical node behavior profiling," UMI Dissertation Publishing, Ames, 2010.

[10] R. Moskovitch, ""Detection of unknown computer worms based on behavioral classi cation of the host," *Science Direct*, pp. 4544-4566, 2008.

[11] W. W. a. T. Daniels, "Network Forensics Analysis with Evidence Graphs," in *The Digital Forensic Research Conference*, New Orleans, 2005.

[12] B. Schneier, "Attack Trees," 8 10 1999. [Online]. Available: http://tnlandforms.us/cs594-cns96/attacktrees.pdf. . [Accessed 01 02 2019].

[13] S. B. a. C. Tyagi, ""Comparative Analysis of Relational And Graph Databases," *International Journal of Soft Computing and Engineering (IJSCE,* pp. 509-512, 2012.

[14] H. C. N. K. J. P. S. Lee, "Managing Cyber Threat Intelligence in a Graph Database: Methods of Analyzing Intrusion Sets, Threat Actors, and Campaigns," *IEEE Xplore,* 2018.

[15] M. C. R. A. E. Hutchins, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains," in *6th International Conference on iWarfare and Security*, USA, 2011.

[16] P. S. a. J. H. A. Yan, ""Substructure similarity search in graph databases," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, New York, NY, 2005.

[17] M. L. D. Z. a. D. S. B. Jie, ""Sub-Network Kernels for Measuring Similarity of Brain Connectivity Networks in Disease Diagnosis," *IEEE Transactions on Image Processing* , vol. 27, no. 5, 2018.

[18] A. A. a. A. N. Mahmood, ""Anomaly detection in electric network database of smart grid: Graph matching approach -- Electric Power Systems Research," *Elective Power Systems Research,* vol. 133, pp. 51-68, 2016.

[19] IBM , "IBM Knowledge Center," IBM, 2012. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.crispdm.help/crisp_overview.htm. [Accessed 2019].

[20] W. Vorhies, "Data Science Central," 26 07 2016. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome. [Accessed 15 08 2019].

[21] Y. L. F. Z. ,. C. Y. a. Y. W. Xiaoling Tao, "Graph database-based network security situation awareness data storage method," 2018. [Online]. Available: https://link.springer.com/content/pdf/10.1186%2Fs13638-018-1309-9.pdf. [Accessed March 2019].

[22] G. C. Angles R, "Survey of graph database models," *ACM Computing Surveys*

(CSUR), vol. 40, no. 1, p. 1, 2008.

[23] P. Cichonski, T. Milar, G. T. and K. Scarfone, "Special Publication 800-61 Rev 2 Computer Security Incident Handling Guide," National Institute of Standards and Technology, Gaithersburg, MD, 2012.

[24] W. Wang, A graph oriented approach for network forensic, Ames: Iowa State University, 2010.

[25] R. R. J. L. Rodriguez, "Mordor Documentation," 2019. [Online]. Available: https://mordor.readthedocs.io/en/latest/.

[26] MITRE , "Round 1 Technique Scope," MITRE, 2018. [Online]. Available: https://attackevals.mitre.org/methodology/round1/scope.html. [Accessed 2019].

[27] MITRE, "APT3 Evaluation Scope," MITRE, 2018. [Online]. Available: https://mitre-attack.github.io/attack-navigator/enterprise/#layerURL=https%3A%2F%2Fraw.githubusercontent.com%2Fmitre-attack%2Fattack-evals%2Fmaster%2FAPT3_Round1_Navigator_layer.json. [Accessed 2019].

[28] FireEye, "Threat Reserach: New Zero-Day Exploit targeting Internet Explorer Versions 9 through 11 identified in Targeted Attacks," FireEye, 27 April 2014. [Online]. Available: https://www.fireeye.com/blog/threat-research/2014/04/new-zero-day-exploit-targeting-internet-explorer-versions-9-through-11-identified-in-targeted-attacks.html. [Accessed 2019].

[29] MITRE, "MITRE ATT&CK Evaluations," 2018. [Online]. Available: https://attackevals.mitre.org/methodology/round1/scope.html.