Decreasing the Mean Time to Respond to Network Security Events with a Novel Configuration of SIEM Software

Blake D. Bryant, Hossein Saiedian, Member, IEEE

Abstract—Today’s information networks face increasingly sophisticated and persistent threats, where new threat tools and vulnerability exploits often outpace advancements in intrusion detection systems. Current detection systems often create too many alerts, which contain insufficient data for analysts. As a result, the vast majority of alerts are ignored, contributing to security breaches that might otherwise have been prevented. Security Information and Event Management (SIEM) software is a recent development designed to improve alert volume and content by correlating data from multiple sensors. However, insufficient SIEM configuration has thus far limited the promise of SIEM software for improving intrusion detection. The focus of our research is the implementation of a hybrid kill chain framework as a novel configuration of SIEM software. Our research resulted in a new log ontology capable of normalizing security sensor data in accordance with modern threat research. New SIEM correlation rules were developed using the new log ontology, and the effectiveness of the new configuration was tested against a baseline configuration. The novel configuration was shown to improve detection rates, give more descriptive alerts, and lower the number of false positive alerts. These improvements lead to a reduction in the mean time required for security analysts to detect and respond to network security incidents.


I. INTRODUCTION

The goal of any network security monitoring solution is timely, accurate and actionable network threat alerts. Such alerts are cited as an axiom of mature security organizations such as the U.S. Department of Homeland Security [1]. Unfortunately, security alerts are often prone to false positives based on sensor location within the network, limitations in their ability to apply advanced rule logic, or the inability to represent complex organizational data hierarchies such as: user accounts, critical computing resources, subnet risk levels, and work hours. Additionally, individual security devices themselves may be susceptible exploitation by savvy attackers, affecting the integrity of data they provide [2]. These limitations result in a multitude of alerts flooding security analysts, or a lack of alerts due to overzealous alert suppression [3][4].

Recently, leading information security companies have developed specialized correlation software designed to aggregate data provided by disparate sensor feeds, thus enabling holistic analysis of all network data from a single, centralized, alert feed. Analysis of data from these devices may reveal patterns of activity conducive to fingerprinting individuals or threat groups, based on a trail of data spread across an entire network of sensors. However, it is the development of custom algorithms designed to analyze this data in the context of phased attack ontologies that truly provide additive value in threat detection and prevention.

Additionally, there is a trend of chatty data feeds, such as firewall logs, drastically outnumbering more forensically valuable data feeds such as endpoint operating system logs. Many security experts argue that weeding through troves of firewall log data is impractical and often must be combined with data from other sources for attack attribution. Unfortunately, merely aggregating data from sensors does not greatly improve detection rates nor decrease false-positive ratios.

Discerning notable security events from log data, and implementing timely remediation for incidents, is a daunting task without an effective alerting engine employed to filter, categorize and escalate security events appropriately. Security data must be normalized into a standard ontological framework, analyzed within the context of known attacker methodologies, and finally allowed to accrue suspicion dynamically as threat activity progresses throughout the network to fully realize the axiom of timely, accurate and actionable alerts.

A. Significance

Advanced correlation software in SIEM systems is designed for real-time alerting of potential security events, as well as to increase the investigative and data retrieval functions associated with those events. Analysis of raw sensor feeds is overwhelming for human analysts due to the high volume of alerts and high false positive ratios. Implementing programmatic analysis decreases false positive ratios and provides mechanisms for the abstraction of human labor functions to a higher analytical plane via a unified graphical user interface (GUI). This in turn enables the establishment of analyst pools ultimately improving process efficiency and decreasing the mean time required to triage and respond to net-
work security events.

However, current software solutions for data normalization and threat action modeling within SIEM software are limited. Current solutions merely provide a framework for normalizing disparate data feeds and performing logical comparisons of the metadata contained therein. These tools are often used to implement static trigger criteria based on either volumetric thresholds or watch lists containing threat signatures, but this methodology is prone to false detections.

A method of implementing dynamic suspicion escalation through contextualized data, aggregated from multiple sources, and attributable to specific threat actions is not found within SIEM software by default. A threat framework must first be adopted to attribute malicious activity to specific threat objectives. This framework can then be leveraged to attribute various levels of risk and suspicion according to the extent to which activities satisfy the threat objective phases.

This paper analyzes existing threat frameworks for inclusion within a SIEM solution, with the goal of providing more timely, accurate and actionable alerts through threat attribution and dynamic suspicion escalation. Ultimately, a novel threat model was devised based on the competing threat models evaluated. This novel model was implemented through modifications to the database structure of a commercially available SIEM system.

**B. Research Methodology**

An empirical research methodology was applied to evaluate existing research associated with intrusion detection technology, SIEM software, and network attack methodologies. The concepts of data triage, suspicion escalation, threat ac-tor groups, and models for representing threat methodologies were evaluated. This research lead to the selection of a commercial SIEM product for evaluation of an existing ontology framework used to represent security data in a normalized format. Finally, a laboratory environment was constructed consisting of a security device sensor array, multiple security devices configured in series, and the selected SIEM product. Figure 1 illustrates this laboratory design.

SIEM correlation rules were implemented in accordance with the model devised in this paper. Detection performance was evaluated in relation to a baseline SIEM configuration with vendor recommended correlation rules.

**II. HACKERS AND THEIR METHODS**

**A. Hacker Categories**

It is necessary to understand the nature of threats and threat methodologies in order to establish an accurate ontology framework for alert triage and analysis. Early computer programmers were motivated by friendly competition to establish novel, elegant, or ingenious ways of manipulating technology to solve problems. They used the work “hacking” to describe an iterative discovery process of multiple failed attempts followed by minor improvements, similar to hacking down a tree with an axe. Those who worked to gain unauthorized access were called “crackers,” because of the similarity of their work to criminal activities like cracking a safe [5].

Hald & Pedersen established a taxonomy of hacker groups consisting of nine primary categories: Novice (NV), Cyber-Punks (CP), Internals (IN), Petty Thieves (PT), Virus Writers (VW), Old Guard Hackers (OG), Professional Criminals (PC), Information Warriors (IW), and Political Activists (PA) [6]. These categories are plotted by motivation and skill level in Figure 2.

The Hald & Pedersen categories have overlap in both motivation and skill level. Additionally, the goal of indefinite persistence within a network for future exploitation is not addressed. This paper combines these nine categories into the three categories: prestige, publicity, and profit. Additionally, the fourth category “persistence” is added to address the actions exhibited by modern information warriors.

1) Prestige Hackers

Prestige hackers are most similar to the original hackers, and often focus on developing novel code or techniques with the intent of furthering the computer science, electrical engineering or networking bodies of knowledge. This category combines the Hald & Pedersen categories of novice and old guard. Though benign in their intentions, they may discover vulnerabilities, exploits, techniques or tools that are later employed by more malicious groups. Prestige hackers do not apply a methodical doctrine toward breaching security, but rather in depth analysis of specialized portions of the system.
IEEE Transactions on Information Forensics and Security

2) Publicity Hackers

Publicity hackers are often referred to as “hacktivists,” i.e., hacker and activist, based on the nature of their activities [7]. They focus on defacing publicly visible information as-sets to manipulate media coverage, often in conjunction with ideologically relevant events. Hald & Pedersen cyber-punk and political activist groups are included in this category. The attacks vary in sophistication, but they are usually covert in inception and overt in execution. Sophisticated hacktivists will attempt to avoid detection until their activities are revealed for propagandist exploitation. Unsophisticated hacktivist actions may require no obfuscation, such as an overt denial of service campaign against a public facing website.

3) Profit Hackers

Profit hackers focus on malware development to manipulate information security breaches for financial gain. Virus writers, petty thieves and professional hackers belong in this category. They are more apt to follow an established, or automated, methodology in order to reap the benefit of economies of scale. Victims will be targeted indiscriminately and techniques will be reused multiple times to affect the largest number of systems possible and increase the potential for profitability. They will continue to use well known tools or techniques as long as they remain effective, and they consider the volume and rate of system compromise to be more important than avoiding detection.

4) Persistence Hackers

Persistence hackers are the most dangerous and difficult category to detect. This category includes corporate espionage and nation state actors, and the Hald & Pedersen in-formation warrior and internal threat groups. Their primary goal is to breach network security and maintain a persistent presence within the targeted environment to gather information indefinitely. They will employ a sophisticated and methodical approach to network penetration and will avoid reusing tools that have been detected previously. Actions are designed to appear like routine network traffic and remain below the detection thresholds of individual sensors. The remainder of this paper will focus on persistence hackers techniques and detection.

B. Hacker Methods

Recent research on network intrusion detection systems, and security event management systems have existed for several years; however, most research focus on technical challenges associated with data analysis rather than psychological motivations of attackers [8][9][10][11][12][13][14][15][16]. An ontological framework representing how persistent threat groups penetrate networks and exploit vulnerabilities is seldom addressed in contemporary research on intrusion detection.

Recently the term “kill chain” emerged within discussions pertaining to information security and network detection. The term kill chain is derived from the Department of Defense joint targeting process [17], which was designed for the positive identification and attribution of culpability to actions associated with suspected actors. The US targeting kill chain is epitomized by the acronym F2T2EA, which consists of six phases: Find, Fix, Track, Target, Engage and Assess. This is similar to a pipe and filter model in software engineering, with the product of one phase providing input to subsequent phases in a serial fashion. Disruption of any phase within this chain will result in the dissolution of the process in its entirety. The following studies outline kill chain model research.

1) The Lockheed Martin Intrusion Kill Chain

The most prominent model associated with the term “kill-chain” within network security research is the Lockheed Martin Intrusion Kill Chain. In this model, Advanced Persist-tent Threats (APTs) employ a methodical targeting process similar to the DoD kill chain [18]. Lockheed Martin’s “intrusion kill chain” describes the seven phases of activities APTs conduct to compromise a system: Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command and Control (C2), and Actions on the Objective. However, the Lockheed Martin model does not adequately address actions other than data exfiltration which can occur after a persistent threat has compromised a system, such as lateral reconnaissance to determine more susceptible systems, followed by repetition of phases one through six. It is uncommon for an advanced threat to attempt to transfer data from the initial compromised system, since such activity in-creases suspicion and may risk loss of the system as a persistent access point into the network.

2) Mandiant APT Attack Lifecycle Model

The Mandiant Corporation devised a six phase model called the “Mandiant APT Attack Lifecycle,” which includes the iterative process attackers employ to gain additional footholds within a network following the initial compromise. This model considers the possibility of branch and recursion at phase five, spawning sub-phases associated with lateral infection [18].

The Mandiant model greatly simplifies the initial phases of the Lockheed Martin kill chain by incorporating the weaponization, delivery, exploitation, and installation phases into a single phase called initial compromise. The Mandiant model also labels phases based upon intent rather than action, helping to aggregate actions that serve a common purpose.

Another key differentiator between the Lockheed Martin model and the Mandiant model is the escalate privileges phase. Mandiant identifies multiple tools used by APT groups to gain access to additional resources on the compromised system, which provide behavioral signatures that may serve as key indicators of compromise and differentiate between routine and persistent threat activity.

The APT may then continue to infect additional systems by progressing to phase five and its sub-phases, or it may culminate with phase six. These optional recursive phases consist of network reconnaissance to identify additional prospective infection vectors. Unlike the initial reconnaissance phase, this is almost exclusively network-based and may be detected by anomalous network traffic associated with internal scanning.

Though the actions will appear very similar to previous phases, their purpose and sequence differ. As mentioned previously, a sophisticated attacker is unlikely to leverage their initial foothold for launching their final attack. There-fore, differentiating the initial foothold from persistence activity is
an essential part of an investigation and should be considered when developing a monitoring solution. Likely indicators of persistence within a network include backdoor software being installed on additional systems and/or the establishment of a covert channel for persistent external data transmission. If the attacker chooses to implement the optional recursion phases and maintain presence through lateral infection, privilege escalation will be conducted on subsequent machines to establish persistent access. This provides another opportunity to gather data associated with the compromise.

The Mandiant model improves the granularity of events along the kill chain which may indicate compromise, but it does not make direct applications to SIEM technology, intrusion detection algorithms, or rule generation.

III. SECURITY INFORMATION AND EVENT MANAGEMENT (SIEM) SOFTWARE

A. Origins

Amrit Williams and Mark Nicolett coined the term SIEM in 2005 to describe the convergence of Security Event Management (SEM) and Security Information Management (SIM) software into a single consolidated product [20][21]. Historically, SIM software was focused on post-incident review and analytics, while SEM software was designed to provide real-time alerting of intrusions or other security incidents. SIEM products additionally provide log management services since log collection, analysis, and retention are integral parts of the SIEM process.

Several papers have been written to address individual components that provide data for SIEM systems, such as improving detection ratios in low level sensors [22], log retention and management data structures [23], and packet inspection [24]. However, few studies have been conducted on SIEM software and its underlying mechanisms: security event management, threat taxonomies, attack ontologies, and incident weighting. Understanding these mechanisms provides insight to potential areas for optimization.

B. SEM Data Triage, Analysis and Ontologies

SEM systems focus on the process of actively detecting security events as they occur. The following SEM models established the theoretical basis for future SIEM systems.

1) Progression of Dangerousness

Legrand addressed the task of wading through holistic network analysis alerts by subjecting normalized SEM data to a static causal event ontology based on five factors: why, who, where, how and what [27]. Each ontological factor must be satisfied by an observable network event and the summation of these events constitutes an action. The result of this ontological analysis is run through a threat algorithm called the progression of dangerousness, where actions are weighted to identify which are the most threatening to net-work assets. Action weighting is calculated via the function f(a) = (d1(a),d2(a),…,dp(a) where each observable action ‘a’ is iteratively evaluated against all ontological dimensions d1 through dp. This model relies heavily on the intrinsic detection capabilities of sensors.

Chien et al. proposed a two-layer attack framework where Primitive Attack (PA) sensor information feeds into an attack subplan layer, based upon attack subontology and attacker intent [28]. The ontology has three classes: reconnaissance, penetration, and unauthorized activity. This signifies the transition from static ontological analysis to a dynamic ontology with classes dependent upon the state transitions between PAs. Chien also introduced the notion of assigning confidence values to detections on a per sensor basis. Chien’s primitive attack layer expands upon Jingxin’s event verification module concept as well as incorporating Legrand’s concept of ontological integration. Higher level subplan templates are used to align disparate PA information into a coherent attack based on known or suspected attack methodologies.

2) Visualization and Graphical Tools

SIEM software may be improved with visualization tools for postmortem incident auditing and predictive analysis. Kotenko and Novikova outlined the essential functions of a SIEM visualization subsystem: Real time data monitoring, integration with a historical data repository, graphical inter-face for rule editing and generation, attack modeling, and resource management [29]. Histograms, linear diagrams, and dashboards are all useful.

3) Filtering and Correlation

Flynn focused on implementing kill chain methodologies in SIEM software and stressed collecting event data on routine activity so holistic analysis may be conducted on security incidents [30]. A continuum of progressive suspicion is needed, similar to Legrand’s progression of dangerousness. Flynn proposed an “event pipeline” framework consisting of blacklisting, identity translation, correlation, context, and analysis. Blacklisting in this context is the removal of known false positives, such as those which match signatures stored on intrusion detection systems, but which are associated with operating systems that are not in the network. Identity translation entails maintaining a record of internal machines, users, and IP addresses for future correlation. Correlation has two sub-phases: the attack plane and the kill chain [30]. The attack plane compares disparate events with some shared identifying characteristics to determine group events for context and suspicion escalation. The Lockheed Martin model is the basis for the kill chain, which provides criteria for attack plane grouping. Context is the fusing of external information surrounding the detection, such as cross-referencing network diagrams. In Analysis, a correlated and contextualized alert is provided to a human for review.

IV. DEVELOPMENT OF A NOVEL SIEM CONFIGURATION: REVISITING THE LOG CLASSIFICATION ONTOLOGY

A. Overview

Evaluation of data derived from real world penetration tests indicated a strict hierarchical model of chained events, described by many of the preceding models, may not be feasible in systems consisting of data aggregated from multiple
disparate subsystems. This was primarily due to instances where essential data for strict correlation sequencing was either missing or omitted by design. However, consistent recurring patterns of evidence emerged within similar phases of an attack lifecycle. A framework representing different attacker objectives, tasks, and related forensic data was created to serve as a new SIEM log ontology based upon these observations, deviating from either of the “kill-chain” models described previously [31].

B. Investigation Framework

Attacker activities observed across multiple networked systems occurred within four logical domains: network, endpoint, domain, and egress. Figure 3 shows the relationship between these phases and the hybrid kill chain model.

![Network Phase - Endpoint Phase - Domain Phase - Egress Phase]

Fig. 3. Investigation framework phases derived from the four logical domains of: network, endpoint, domain and egress.

1) Network Investigation

The network domain consists of data that is often provided by network devices, such as routers, switches, remote access devices, network scalers, intrusion detection systems, and firewalls. Data provided by these devices typically contains the following metadata: IP address of origin and destination devices, port numbers, and a signature. The payload or packet capture analyzed by security devices may be pro-vided, but is not guaranteed. This data may also be correlated with data provided by devices monitoring the endpoint and domain phases, if the logged data is associated with network activity. The network phase consists of two objectives: reconnaissance and delivery.

The attacker objective of reconnaissance is further divided into probing and enumeration. Probing detects live hosts on a network and maps to the host identification techniques in the penetration test data. This will often include ICMP traffic sent to sequential IP addresses as well as crafted TCP and UDP packets sent to common service ports. TCP scan packets will often be sent with a SYN flag only, with the at-tacker observing a SYN/ACK response from the probed host without sending an ACK flag to complete the connection. Enumeration consists of operating system fingerprinting and service discovery. Whereas probing may determine that a specific service such as email is being hosted on a server, enumeration is used to determine which version of software is being utilized. This is often done through a process referred to as banner grabbing. Figure 4 outlines some of the indicators associated with reconnaissance activity.

The attacker objective of network delivery is further divided into host access and payload delivery. Host access is accomplished when an attacker authenticates to a service running on an endpoint. This may manifest as a remote terminal session, or a successful response from a vulnerable service such as a DNS zone transfer. Successful service authentication indicates that it may be possible to transfer a malicious payload to the endpoint. Payload delivery is at-tempted following the identification of a vulnerable service channel in the host access phase. However, novice attackers may attempt to deliver a payload without executing the reconnaissance or host access phases, which generates a large volume of intrusion detection system alerts. This phenome-non explains the large number of false positives observed in intrusion detection system alerts, as the payload may not be confirmed as an effective network attack unless the end-point is running the vulnerable service the attack was crafted to exploit. Therefore, indicators observed within the network phase do not necessarily indicate a system has been compromised, but merely that an attacker is searching for holes in the system. Figure 5 illustrates observable data associated with delivery activities.

2) Endpoint Investigation

The endpoint phase consists of data extracted from logs stored locally on a computer, such as a workstation or server. Data provided at this level consistently provides the computer name of the device logging the activity as well as data specific to the actions performed. Network related actions may provide the IP addresses of origin and destination devices and port numbers. Application modifications will provide vendor specific signatures or messages. Authentication or privilege use will provide account credentials and indicate the level of privilege granted at the time of use. These logs are especially useful in discovering unauthorized software installations via application whitelisting. Additionally, this data may provide insight to the tools or commands used by an internal attacker. The endpoint phase consists of two attacker objectives: installation and privilege escalation.

The attacker objective of installation consists of two subordinate tasks: host delivery and software modification. The host delivery task is similar to network delivery discussed in the network phase, but the detection mechanism and content of logs in this phase differ from the network phase. Anti-malware products are the most likely mechanisms to detect the presence of malicious code uploaded to an end-point. This data may corroborate data detected in the network phase, or identify payloads that avoided detection by network intrusion detection systems. Malicious code may also be identified by monitoring endpoint files and by folder integrity monitoring via operating system audit logs. Whereas the previous network phase may have reported attempted but unsuccessful payload delivery, this phase confirms the presence of malicious software on the endpoint. The software modification task involves the installation or registration of malicious binaries, or the modification of existing software to serve a malicious purpose. Registry key modifications, file or folder access, scheduled task registration, service registration and starting, and Windows installer logs are useful in detecting this type of activity. Figure 5 illustrates common indicators of installation activity.

The attacker objective of privilege escalation consists of two subordinate tasks: privilege escalation and privilege use. The privilege escalation task is associated with gaining administrative access on an endpoint. This may be represented as direct security group manipulation, such as creating or modifying a security group, or it may be represented as
credential replay, such as passing a hash. The privilege use task is represented by evidence of administrative level actions exercised on an endpoint system.

This may be observed by the endpoint reporting an administrative logon during authentication or via a “run as” command, wherein credentials other than those of the current account are used to execute commands at a higher privilege level. Figure 6 illustrates data observed during privilege escalation actions.

3) Domain Investigation

The domain phase consists of data that resides on the central authentication server, typically a domain controller in a Microsoft Windows domain. This data consistently contains the computer name and account name associated with observed activity. Network data, such as IP address and port numbers may also be provided for remote authentication. This data is beneficial for detecting unusual communication between internal computers, or by accounts that do not often communicate with specific devices or directories. There may be similar or redundant data logged by the domain controller and the local machine manipulated by an attacker during this phase. As such, logs stored on the domain controller may be compared to logs stored on local machines to detect attacker attempts to destroy evidence and avoid detection. The domain phase consists of lateral movement and actions on the objective.

The attacker objective of lateral movement consists of two subordinate tasks: internal reconnaissance and lateral movement. Internal reconnaissance is similar to reconnaissance observed during the network phase, however it is often conducted from an internal host rather than from the attacker’s original machine, so legitimate processes in the compromised operating system may be used to avoid detection by anti-malware software. Since local processes are used, there are additional opportunities for forensic data both on the compromised endpoint and the domain controller logging attempts between internal hosts. In situations where organizations haven’t deployed intrusion detection systems, endpoint or domain controller logs may be the only systems providing forensic data of interest. Lateral movement is the process of using compromised credentials and privileges on additional internal hosts within the network. Domain controller logs provide the unique ability to track privilege use across disparate endpoints. Figure 8 illustrates indicators of possible lateral movement activity.

Attacker actions conducted in the “Actions on Objective” phase are associated with actualization of the attacker’s primary purpose for infiltrating a network. This stage normally consists of data manipulation, such as copying data, deleting data, or modifying permissions. Additionally, experienced attackers will likely attempt to modify system logs or security devices to remove evidence of tampering conducted within this phase. Figure 9 illustrates potential indicators observed during the actions on the objective phase.

4) Egress Investigation

The egress phase is identical to the network phase in regards to data provided by monitoring devices, but this phase is differentiated by the direction of travel and presence of known malicious actor indicators, such as blacklisted IP addresses, domains, or email addresses. This phase may be an indicator of compromise even if indicators were not observed in previous phases. Figure 10 depicts common indicators observed during the data exfiltration phase.
Fig. 6. These are common indicators observed during the installation phase when an attacker attempts to modify a target operating system.

Fig. 7. Common indicators associated with attacker attempts to escalate privileges on a targeted system are depicted above.

Fig. 8. Attacker attempts to gain access to multiple devices within an internal network may be detected via indicators listed above.

Fig. 9. The actions on objective phase may be identified by some of the indicators depicted within this figure.
V. APPLYING THE KILL CHAIN TO SIEM SOFTWARE

The LogRhythm® commercial SIEM platform was selected as the preferred system to evaluate inclusion of a kill chain model based on the author’s prior experience with the system and access to historical data conducive to evaluating multiple production environments. The IBM Qradar, McAfee Nitro, and Splunk platforms also exhibited potential to be modified to incorporate this model, but were not evaluated within this paper.

The LogRhythm data flow model, depicted in figure 11, implements suspicion escalation and data triage functions by parsing sensor information into a threat ontology and applying descriptive classification labels to observed events. The classification label is potentially applied in two different stages of the data flow model; either during the initial parsing and normalization phase by the message processing engine within the log manager, or by the advanced intelligence engine during correlation and subsequent reclassification.

Classification labels serve a unique function as they introduce new metadata into an event record that was not present within the raw log information. This provides a mechanism for combining previously dissimilar data from disparate sensors into corroborating data sets. The classification field was determined to be the ideal candidate for implementing the new kill chain phased model within the SIEM, since this would facilitate rapid identification of related events and enable future event correlation during alarming.

Deconstructing the hybrid kill chain and analyzing metadata within each sub-phase yielded insight to potential data pairings for correlation. Each sub-phase was evaluated for suitability as a table within a relational database. Metadata within each phase was evaluated for suitability as primary or foreign keys to be used to join adjacent sub-phase data as the LogRhythm® SIEM utilizes SQL queries to perform correlation functions.

Some logging systems did not provide enough data for correlation with adjacent phases without fusing data with another source within the same phase. Correlating network delivery events with installation events is a prime example of this phenomenon. Network intrusion detection systems often omit the hostname of machines, while host-based malware solutions often contain the hostname but omit the local IP address. This issue was resolved by fusing both data sets via DHCP, DNS, or domain authentication data available on domain controllers or servers hosting these common services.

Sensor logs, SIEM events, or security alarms were aggregated within each phase with SQL queries joining metadata via classification field and the phase-specific aggregate field. For example, all events classified as being associated with the reconnaissance objective group (reconnaissance, probing, or enumeration) with the same source IP address were aggregated within a single event. The aggregate event retained all unique metadata fields observed within aggregated records to provide the maximum forensic value to analysts.

VI. TESTING AND EVALUATION OF THE NOVEL SIEM CONFIGURATION, PART 1: LABORATORY AND EXPERIMENT DESIGN

A sophisticated network security laboratory environment was designed to evaluate the efficacy of the SIEM configuration modified with a novel ontology. Two identical laboratory environments were constructed with the single variable between deployments being modifications to the SIEM database used to detect security events. This section focuses on the design of the laboratories and the details of the experiment for which they were used.

A. Laboratory Network Design

A virtual network was constructed to evaluate baseline and enhanced SIEM configurations. Two separate but identical virtual environments were constructed, with the exception that the LogRhythm® SIEM system in one environment was configured with vendor recommended default correlation rules and the other environment contained a LogRhythm® SIEM system enhanced with additional classification fields reflecting the hybrid kill chain model.

Microsoft operating systems were selected as the basis for the majority of virtual systems within the laboratory network due to security analysts’ familiarity with conducting forensics investigations based on Microsoft technology as well as a more robust library of default SIEM correlation rules designed for Microsoft systems. Services hosted on Microsoft systems included: directory services, email hosting, web services, and a SQL database. A suite of McAfee antimalware products was deployed to endpoints to provide antivirus and host based intrusion prevention system data via a centrally managed...
server. A pfSense virtual machine was deployed to serve as a virtual layer three device, necessary for network traffic shaping, as well as a platform to host open source security tools including: Snort IDS, Squid proxy, and network based firewall capabilities. All Microsoft endpoints were configured with host-based firewall settings to provide an additional layer of security beyond network based filtering as well as provide supplementary data for correlation with data provided by network centric sensors. Audit policy settings on all Microsoft endpoints were adjusted to provide additional forensic details omitted by default configuration settings, such as logging network traffic denied by host-based firewalls or process creation.

B. Attack Experiment Design

Real world security breaches do not always reflect every stage represented by the hybrid kill-chain model. As such, a custom scenario was devised to stimulate sensors and ensure coverage of all seven stages. This scenario combined traditional reconnaissance and probing techniques, indicative of opportunistic attacks, as well as targeted attacks typical of advanced persistent threats. Table 1 presents a list of test cases associated with the attack scenario as well as comparative detection rates between baseline and enhanced SIEM systems.

<table>
<thead>
<tr>
<th>Case Name</th>
<th>Base SIEM Alarm</th>
<th>Base Event</th>
<th>Mod SIEM Alarm</th>
<th>Mod SIEM Event</th>
<th>Raw Logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nmap port scan</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>100</td>
<td>87</td>
</tr>
<tr>
<td>SMB scan</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>OpenVas scan</td>
<td>41</td>
<td>41</td>
<td>5</td>
<td>401</td>
<td>4158</td>
</tr>
<tr>
<td>Phishing email</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>92</td>
</tr>
<tr>
<td>File download</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>Software install</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>18</td>
<td>105</td>
</tr>
<tr>
<td>Reverse shell</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>344</td>
</tr>
<tr>
<td>New local admin on workstation</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>997</td>
</tr>
<tr>
<td>Remote desktop session</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>174</td>
</tr>
<tr>
<td>Disable ant-virus</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>86</td>
</tr>
<tr>
<td>Meterpreter shell</td>
<td>18</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>106</td>
</tr>
<tr>
<td>Hash extraction</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>55</td>
</tr>
<tr>
<td>Mount network share</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>33</td>
</tr>
<tr>
<td>Internal recon tool use</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>54</td>
</tr>
<tr>
<td>Pass the hash from Kali Linux to web server</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>27</td>
<td>80</td>
</tr>
<tr>
<td>Copy SQL database</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>250</td>
</tr>
<tr>
<td>New local admin on web server</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>61</td>
</tr>
<tr>
<td>RDP to web server from workstation</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>11</td>
<td>353</td>
</tr>
<tr>
<td>Transfer data from webservice to workstation</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>Pass the hash from Kali Linux to email server</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>New local admin on email server</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Copy email database</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>12</td>
<td>131</td>
</tr>
<tr>
<td>RDP from workstation to email server</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>204</td>
</tr>
<tr>
<td>Transfer data from email server to workstation</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>Transfer data from workstation to Kali Linux</td>
<td>18</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>56</td>
</tr>
<tr>
<td>Clear audit logs</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>11</td>
<td>304</td>
</tr>
</tbody>
</table>

VII. TESTING AND EVALUATION OF THE NOVEL SIEM CONFIGURATION, PART 2: EXPERIMENT RESULTS

A. Detection Rate Comparison

The modified SIEM ontology outperformed the baseline SIEM ontology in alert metrics with a 96% true positive detection rate by generating an alert for 25 out of 26 test scenarios. The baseline SIEM ontology and LogRhythm© default rule set had a 26.9% detection rate with alerts generated for 7 out of 26 of the test cases. Additionally, the modified ontology generated aggregate alerts with metadata from multiple events for 76% of alerts (19 of 25). The remaining six alerts were associated with singular events where no additional data was available for aggregation.

It is worth noting the difference in alert volume in addition to improvements in the true positive rate. The baseline SIEM generated 83 alerts during the evaluation, however they were only associated with 7 of 26 test cases. An Open-Vas vulnerability scanner test case resulted in nearly half of the baseline SIEM alerts with 41 separate alerts. Conversely, the modified SIEM generated 5 alerts during the same test case, containing aggregate metadata from 401 correlated events, and 46 alerts from all test cases. This data indicates the ability to aggregate data via a logical identifier metadata field proved to be an effective mechanism for decreasing alert volume.

A detailed comparison of alerts between the baseline and modified SIEMs are listed in table 1.

B. Alert Forensic Value Comparison

The primary motivation for developing the new SIEM ontology was to provide a mechanism for the aggregation of pertinent and related metadata into alert notifications to decrease the investigative effort associated with explaining security alerts.

The baseline SIEM ontology combined 47 OpenVas test case alerts into a single email containing 7,154 words. It was not obvious which metadata field was used to correlate these events, since none of the fields were common across all 47 alerts. The email batching process merely listed alerts, rather than combining them in a logical manner. Only 41 alerts were generated within the analyst GUI console during the OpenVas scan test case, indicating six additional alerts must have been aggregated from previous scan activity. It appears this aggregation was most likely performed based on the large increase in alerts generated within a short time frame during the scan, resulting in combination based on temporal proximity, rather than through metadata correlation. Many of the alerts in the batch of 47 alerts generated during the OpenVas scan correctly identified abnormal net-work connections to the Windows 7 host W7host with IP address 10.13.201.94, as was replicated during the scan; however, no additional information was provided to indicate which computer(s) were attempting to communicate with the workstation, nor what aspect of the communication was considered abnormal. An analyst would be required to re-view all 47 alerts generated in order to identify the attacking machine or the scope of the probes conducted within the network.

Ten of the alerts in the pool of 47 correctly identified the attacker machine as the origin host with IP address 172.16.0.3, but it was not obvious what actions this host was conducting...
within this batch of alerts. One alert indicated a machine with IP address 172.16.0.3 was suspected of being associated with a system compromise or lateral movement, but there were no metadata artifacts associated with the alert to indicate how the conclusion was reached. In reality, the attacker had not yet successfully compromised a machine at this point.

Four of the 47 batched alerts indicated suspicion of a port scan, but only one of these four alerts indicated both the source and destination machines associated with the port scan activity. The remaining three alerts only indicated the targeted machine.

1) Modified SIEM Ontology Email Alert Analysis

In contrast to the 47 batched alerts generated by the baseline SIEM ontology, the modified SIEM ontology accurately identified the scan activity with a single alert. This was achieved by aggregating metadata fields from multiple events within the alert. The event field within the alert shows that 92 related events were combined. All alert notifications generated in the baseline configuration were comprised of a single event, even when batched. The modified SIEM alert title, depicted in the email subject line, identified the event as being associated with suspected reconnaissance activity and the aggregate field for correlation was the origin host field. The origin host, was correctly identified as the Kali Linux machine with IP address 172.16.0.3. The entire list of targeted machines was provided within the alert. Supporting metadata, including port numbers and names for aggregated events, were also provided.

2) Alert Forensic Value Conclusions

The modified SIEM alerts provided considerably more correlated data than the baseline SIEM alerts. As a result, security analysts were more likely to receive enough information to draw conclusions regarding the nature of the activity, requiring fewer manual queries to validate their hypothesis. The alerts presented using the baseline SIEM configuration often required a considerable amount of analysis of similar alerts to determine what data was actually detected and what data may warrant additional investigation. From a forensic perspective, the data contained in the modified alerts was superior to the data contained in the baseline SIEM alerts.

C. Email Alert Volume Comparison

The baseline SIEM configuration generated 2,364 alerts from 9/29/2015 to 10/21/2015, averaging ~100 alerts per day. Conversely, the modified SIEM configuration generated eight alerts from 11/23/2015 to 11/30/2015, averaging one alert per day.

The decreased alert volume may be attributed to the decreased number of detection rules configured between the two deployments. The modified SIEM had less than a third of the rules of the baseline SIEM, and 99% fewer alerts when test data was not being generated. The baseline SIEM rules generated an average of .78 alerts per rule per day, while the modified SIEM rules generated an average of .025 alerts per rule per day. In light of the modified rule set’s improved true positive detection rate, it is determined the decrease per alert rule rate during non-testing conditions reflects a decreased false positive rate.

D. SIEM Rule Complexity Comparison

The baseline SIEM rule set consisted of 128 correlation rules while the modified SIEM rule set consisted of 39 rules. This was achieved by segregating rules into separate groups consisting of specific event queries and aggregate alarm queries, while the baseline SIEM configuration used only specific queries. The decreased number of queries required to detect threat actions is assessed to be an improvement over the base model due to the assumption that fewer administrative actions will be required by SIEM engineers to maintain the system. Additionally, the queries contained within the modified SIEM rule set hierarchy were generally less complex than the baseline rule when compared side by side.

1) Baseline SIEM Rule Complexity Analysis

Many baseline SIEM rules consisted of two stages, a baselining or learning stage and a threshold comparison stage. The baselining stage constructs a dynamic list of unique values during the learning period, which is configured to be seven days by default, and generates an average number of unique values observed by host. The threshold stage searches for deviations from the baseline. One such rule searched for more than five unique processes running in memory beyond the average determined by the baseline. This rule consumed approximately 17% of the memory allocated to the Advanced Intelligence Engine service running on the SIEM.

2) Modified SIEM Rule Complexity Analysis

Modified rules leveraged static lists of indicators derived from the investigations conducted earlier, rather than a baselining mechanism. Using a static indicator list removed the need to implement a multi-stage rule of baselining and threshold establishment. For example, a modified rule designed for detecting rogue processes was configured to generate an event within the SIEM event database for any process name observed but not listed on the static list of approved processes, however this would not generate an alert by itself. Alert threshold establishment was performed by separate rule blocks that would aggregate events or alerts with similar classification tags and predesignated metadata aggregate fields specific to each kill-chain phase. This configuration allowed every violation to populate an event database for investigation by analysts even if no alert was triggered. The baseline configuration only generates an anomalous event if the threshold is exceeded. The memory resources consumed by the modified SIEM query were negligible and reported as 0% of the total resources available to the Advanced Intelligence Engine process running on the SIEM. This is a marked improvement over the baseline SIEM rule constructed to per-form the same function.

E. Investigation Framework Analysis

A pool of security analysts was selected to implement the hybrid kill-chain model to guide network forensic investigations as a mechanism to verify suspected benefits of a methodical approach to data triage and focus on analysis. Benefits noted from this evaluation include:

• Identification of potential false negatives due to data omission or errors in programmatic SIEM correlation logic
• Improved communication between analysts and stakeholders
• Operational process efficiency gains due to reduction in redundant queries.

VIII. CONCLUSION
The modified ontology appears to be an improvement over the baseline SIEM ontology in every dimension measured in this paper. The modifications resulted in a drastic reduction in the number of alerts that provide little forensic value to analysts. Additionally, the amount of data provided on a per alert basis was greatly improved through the novel aggregation mechanism of pairing the modified log ontology classification labels with identity metadata fields specific to each kill chain phase. Though the primary motivation for the modified log ontology revolved around alert forensic value, marked improvements in SIEM resource consumption were noted following the implementation of simplified correlation rule queries. Additionally, it is assessed that simpler correlation queries will result in decreased administrative effort to maintain the SIEM system.

These improvements are assessed to have improved the mean time required to detect security events based on the following factors:
• Increased visibility during network security attacks through improved detection rate (roughly 70% improvement in number of test cases detected).
• Increased number of metadata fields contained within alerts generated.
• Decreased total alert volume.
• Decreased effort required by engineers to deploy detection rules.
• Decreased system resource requirements preventing potential processing bottlenecks.

REFERENCES
Blake D. Bryant, MS, CISSP, MCITP, CCNA, completed his MS degree in information technology at the University of Kansas in spring of 2016. Blake is an assistant professor at the University of Kansas and is currently pursuing his Ph.D. degree in computer science. He has 10+ years of experience in leading computer security organizations (private, including Optiv and Ernst & Young, and military) prior to assuming the role of a professor. Blake is also a current member of the US Army Reserves.

Hossein Saiedian, Ph.D. is currently an associate chair, the director of IT degree programs, and a professor of computer science at the Department of Electrical Engineering and Computer Science at the University of Kansas (KU) and a member of the KU Information and Telecommunication Technology Center (ITTC). His research career includes over 170 publications in a variety of topics in computer science, software engineering, and information security. His research in the past has been supported by the NSF as well as other national and regional foundations. Professor Saiedian is a member of AAAS, a senior member of ACM, a senior member of IEEE.