2015 26th International Workshop on Database and Expert Systems Applications

Analyzing and Predicting Security Event Anomalies:

Lessons Learned from a Large Enterprise Big Data Streaming Analytics Deployment

Colin Puri
Accenture Technology Labs
Accenture LLP
San Jose, CA, USA
colin.puri@accenture.com

Carl Dukatz
Accenture Technology Labs
Accenture LLP
San Jose, CA, USA
carl.m.dukatz@accenture.com

***Abstract*—This paper presents a novel and unique live operational and situational awareness implementation bringing big data architectures, graph analytics, streaming analytics, and interactive visualizations to a security use case with data from a large Global 500 company. We present the data acceleration patterns utilized, the employed analytics framework and its complexities, and finally demonstrate the creation of rich interactive visualizations that bring the story of the data acceleration pipeline and analytics to life. We deploy a novel solution to learn typical network agent behaviors and extract the degree to which a network event is anomalous for automatic anomaly rule learning to provide additional context to security alerts. We implement and evaluate the analytics over a data acceleration framework that performs the analysis and model creation at scale in a distributed parallel manner. Additionally, we talk about the acceleration architecture considerations and demonstrate how we complete the analytics story with rich interactive visualizations designed for the security and business analyst alike. This paper concludes with evaluations and lessons learned.**

Keywords — anomaly detection; batch analytics; data acceleration; D3 visualization; graph analytics; log content analytics; streaming analytics;

I. INTRODUCTION

The opaque nature of modern computing and networking systems makes them vulnerable to cyber adversaries or Advanced Persistent Threats (APT's) presenting an ever growing threat to globally interconnected networks. Many enterprise environments are awash in copious amounts of log files where forensic evidence of those threats and suspect anomalies reside unnoticed in logs until it may be too late. Sifting through the numerous log data sources to find errors and anomalies can be a daunting task but it is critical for activities such as application anomaly detection, compliance, intrusion detection, investigation, and operational intelligence to name a few. Learning the behavior of applications through log traces, understanding the flow of events that occur within many applications, performing analytics at massive scales, and performing analytics with low latency and rapid results with streaming data is of key importance when finding relevant security events and being operationally aware in real-time. To this end, we piloted and deployed a unique system that expands upon our domain knowledge in the corporate, cyber security, defense, and academic realms concerning the application of analytics to uncover anomalies such as intrusion attempts and difficult to detect, surreptitious APT's. Armed with an ever-watching tool, capable of evolving over time and providing context to events, an analyst can be confident that the tool will generate alerts, quarantine and control suspect actors, and stop malicious behavior before irreparable damage occurs to the Enterprise and its assets.

This paper gives an overview of the analytics we utilize, how the application of machine learning is applied to security data to understand typical behaviors, and how security rules are automatically learned and applied to a large scale streaming data set. We lay the foundation of previous works, provide motivation, detail our architecture and algorithms, explain the environmental setup, complete the analytics story with interactive exploration of discovered insights, and provide evaluations. This paper finishes with lessons learned and conclusions drawn from the pilot use case.

A. The Client Pilot

In 2014, we implemented and deployed a pilot of a large security anomaly detection system leveraging a combination of open source and vendor tools at a Global 500 company. The system ingests large volumes of raw log data collected by a Security Information Event Management (SIEM) tool, which collects network security events from all the devices on a network (e.g. routers, switches, servers, workstations, anti-virus, intrusion detection systems, etc.). Our client pilot leveraged analytics to identify low probability and anomalous events. The system is comprised of two technology stacks, which relate to the detection of APT's and contextual anomalies that perform the following:

- **Behavior learning** - learning common behaviors that occur within an Enterprise network and transforming them into probabilistic event graphs (based on extract-transform-load or ETL, distributed storage, distributed processing, and machine learning).
- **Anomaly identification** - understanding why events are more important than others and identifying anomalous events (utilizing machine learning techniques).
- **Real-time anomaly detection** - detecting event chains with highly anomalous attributes based on learned behaviors (uses messaging queues, complex event processing, and in-memory databases).

We would like to thank our Accenture Technology Lab downstream partners, leadership, and colleagues for continuing support during the implementation of this work.

1529-4188/15 $31.00 © 2015 IEEE
DOI 10.1109/DEXA.2015.46

II. Previous Works

Several efforts have been made for detection of anomalies in log contents but many rely on checking log files based on a set of expert-defined rules. Some examples include Logsurfer [3] and SEC [4].

Research has been performed on automatic log analysis that does not require expert input (see [12] and [14] for example); however, limited to no deployment has occurred within the business environments and scalability within the big data Enterprise environments remains a challenge.

Given applications in a variety of domains, including bioinformatics, business process management, and in particular log content analytics, graph analytics has become an important research area [8][6][15] in regards to intuitive understanding of business needs.

Taking inspiration from [6] and [13], which use graph techniques to detect anomalies and are based on how well web search engines acquire content, we apply some of these concepts of graphs for anomaly detection within log traces of network security data as collected and aggregated using existing vendor SIEM tools.

There are products available which offer machine learning techniques for identify anomalous activity in the context of security such as, but not limited to, [7] and [10]. Some security platforms such as [2] are focused specifically on monitoring user behavior and identify when abnormal activity occurs (ex. logging in during off peak hours) while other tools utilize techniques involving compression technology to match event patterns in streaming data [5]. Finally there are tools which store security relevant information in a graph database format by extracting relationships of data present then leverage the information in that graph to make inferences about events that occur [11]. Our work seeks to augment existing tools by providing additional capabilities that do not currently exist in traditional vendor offerings.

There are many existing and competing features from many systems each with pros and cons. The aim of this pilot and proof of concept is not to replicate existing tool features but rather augment the features of existing systems within the framework of a current Enterprise environment.

III. Motivation & Goals

Many security analysts and stakeholders have limited resources and are working on strict timelines, making a need for rapid results with complete situational awareness. Many vendor tools in existence enable security analytics and predominately rely on expert understanding, manual navigation of information, curation of security rules, or may result in many false-positives [9] with little contextual information.

The vast load of data streaming within a corporate network is increasing every day as are the number of security vulnerabilities and exploits; the human security analyst can become quickly overwhelmed and become reactive rather than proactive. Our use case required a proof of concept (PoC) that provided an automated mechanism to ingest information and learn what is anomalous and what is not in the confines of security data. Requirements included a platform agnostic solution, real-time streaming implementation, scalable architecture design, and integration with a heterogeneous environment.

Our goal was to deploy a differentiated technology asset that can effectively capture, learn, discover and provide actionable contextually relevant security information utilizing a data acceleration pipeline. Network traffic patterns are learned, anomalies extracted and graded, and rules are automatically created to inform key security activities for hunter teams in exploration, forensics, auditing, and decision-making. Furthermore, we strove to complete the explanation of security events through example visualizations that increase usability and enable faster insight [1].

IV. Architecture

Through the PoC use case, we research the best approach to solving client needs through implementation of several technology stacks to best measure performance and allow ease of use when implementing the data pipeline. Our methodology for architecture decision-making is based on product research and client experience. One of the challenges constantly encountered when architecting a robust solution is not understanding the core technology options available but also the landscape and technology interactions. The Big Data product landscape has grown saturated with new products all claiming to be “real-time”, “in memory”, “big data”, and “cloud”. Our Data Acceleration project started as a research initiative to help define what products actually do and what combinations clients can use to solve their problems. Our initial study encompassed 70 technologies that focused on addressing three challenges in the data architecture space. There were six key architecture components used to overcome the challenges that decompose into in 14 logical patterns (see TABLE I).

The three challenges that our data acceleration research and prototype addresses are data movement, data processing, and data interactivity and are defined as follows:

- **Data movement** is the transporting of data into a system, which is typically solved with streaming or batch.
- **Data processing** solutions involve using big data technology, complex event processing tools, and appliances to meet processing needs.
- **Data interactivity** is about providing results of analytics as quickly as possible to a user or application.

The methods available to address these three challenges general fall into six categories that our pilot prototype implements:

- **Big data platform (BDP)** – A BDP is a distributed file system and compute engine that facilitates data movement and processing. BDPs contain a Big Data Core (BDC) with distributed data storage, computing power, and often function as a platform for additional computing including data interactivity.
- **Ingestion** - Ingestion is about collecting, capturing, and moving data from sources to underlying repositories where users can process it. Focusing more on data

collection, modern techniques act on streaming data (e.g. website clicks), and involves queues.

- **Complex event processing (CEP)** - CEP is a method of tracking and processing streams of event data (e.g. click streams) from multiple sources to infer and identify patterns that suggest more complicated circumstances.
- **In-memory database (IMDB)** - An IMDB is a database management system that relies primarily on main memory for data storage. IMDBs have low latency with simpler algorithm internals requiring fewer CPU instructions and experience faster seek times.
- **Cache clusters** - Cache clusters are groups of servers acting as in memory layer intermediaries with centralized management software mitigating load from upstream data sources to applications and users.
- **Appliance** - An appliance is a prepackaged unit of hardware, software, and support services built with redundancy to provide reliability often utilizing a common database for online transaction and analytic processing reducing system latency.

Technology features which enable onboarding of data from multiple sources in multiple ways for each architecture pattern are categorized as having enhanced movement options. Stream processing is seen as a differentiator over patterns that only offer methods for batch processing. Technology patterns that offer capabilities to customize data allocation for in-memory with querying is seen as having enhanced interactivity.

TABLE I. Architecture Patterns

Pattern	Movement	Processing	Interactivity
Appliance Only	Enhanced	Enhanced	Enhanced
BDP to Appliance	Enhanced	Enhanced	Enhanced
Streaming to Appliance	Enhanced	Enhanced	Enhanced
BDP Only	Enhanced	Basic	Basic
Streaming to BDP	Enhanced	Enhanced	Basic
BDP - In-Memory Analytics	Enhanced	Enhanced	Enhanced
Streaming to BDP - In-Memory Analytics	Enhanced	Enhanced	Enhanced
BDP with query engine	Enhanced	Enhanced	Enhanced
Streaming to BDP - query engine	Enhanced	Basic	Enhanced
Dist. Cache Cluster only	Enhanced	Enhanced	Enhanced
BDP to cache cluster	Enhanced	Basic	Enhanced
IMDB Only	Enhanced	Basic	Enhanced
BDP to IMDB	Enhanced	Basic	Enhanced
Streaming to IMDB	Enhanced	Basic	Enhanced

The main takeaways from TABLE I are:

- CEP can enhance streaming ingestion
- CEP can increase speed by pre-processing data
- Caches and IMDBs enable real-time interactivity

The solution landscape depicted in Fig. 1. decomposes into the 14-high-level patterns that are enumerated in TABLE I and serves as starting points for data ingestion, storage, and processing technical solutions. Our implementation relied on complete product testing with machine learning leveraging SIEM data from HP ArcSight. Over the course of the pilot

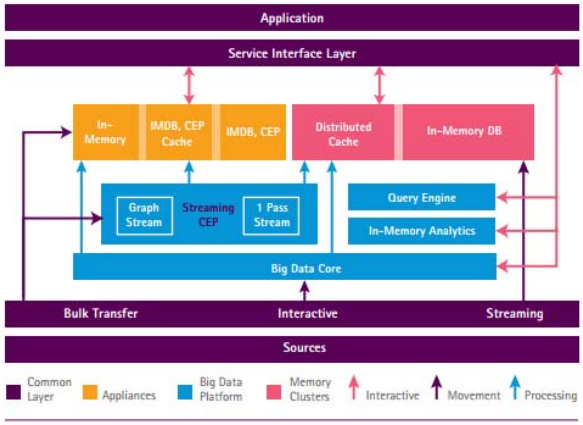

Fig. 1. Solution Landscape

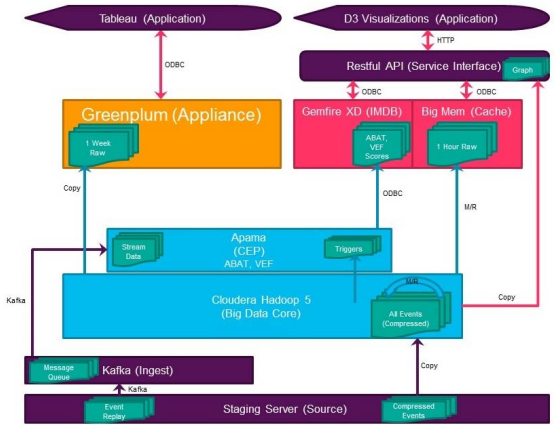

Fig. 2. Prototype Logical Architecture

PoC, we collected approximately 3PB's of log information from a large Global 500 client with the objective to enable Security Operations Centers and Incident Response Teams with contextual information on anomalous events occurring on the network with respect to risk. Leveraging a lambda architecture for storage and processing, machine learning was executed over Hadoop and the resulting graph analytics were loaded into a CEP engine with new events streamed via a messaging queue (see Fig. 2.).

V. Analytics

Our log content analytics (LCA) graph learning framework resides atop our data acceleration pipeline and applies analytics to (semi-) automatically consume and analyze heterogeneous computer generated security relevant trace entries. Characteristics of data present in the contents of trace events and log files include unique identifiers, timestamps, events, and actions. These unique attributes can be indicative of underlying behaviors, processes, and patterns that exist through a series of events from a network agent. The LCA models a rich graph with descriptive statistics, event states, and transition probabilities between events through mining the information contained within a set of traces. Upon establishing a master graph of normalized data, we can analyze incoming data for anomalies

using graph matching, pattern recognition, or other techniques to extract information.

A. Graph Mining Framework

The LCA has several layers for data collection, data ingestion (agnostic normalization, parsing, & storage), data querying, mining, analytics, filtering, and an API wrapper allowing for extensive use and interplay with other tools and visualization applications (see Fig. 3):

- **Data collection layer** - This layer controls the access and instruction set necessary to enabling collection of data between an operational log management (OLM) provider and LCA framework.
- **Log ingestion module** - This layer ingests data, parses it, and normalizes as well as manages stored models.
- **Query module** - This query module is a mediator into lower modules for higher modules abstracting away data format intricacies.
- **Data mining module** - This module performs extraction and discovery of relationships within the data via a plugin framework.
- **Analytics module** - This module performs the analysis of any data that has been mined and extracted, e.g. event clustering.
- **Filter module** - This module performs the filtering of any data analytics and singles out information based on predefined metrics.
- **Command/API dispatcher** - This module serves as an orchestration layer between the command line, lower layers, and web service.
- **Web Service Layer** - This module exposes the functionality of the framework to external tools.

The current framework allows for the application of any number of mining analytics and filtering through extensible plugins. Each of the individual mining algorithms executes in parallel through the distributed paradigm of Hadoop Map-Reduce using Yelp's MRJob. Information flows from one mining algorithm to the next, spinning up new Map-Reduce jobs as necessary. Multiple mappers are instantiated that perform normalization, mining, and model creation before model aggregation at a reducer stage.

The preprocessing stage consists of the selection and ingestion of trace event information into a query-able format for normalization. Once normalized, the LCA framework proceeds to mine the log traces for their placement in a behavior graph model. The mining algorithm extracts an event type and an identifier for each ingested trace entry. The framework indexes the event and identifier pair to create a series of compound event sequences. Additionally, the algorithm discovers and sorts the temporal relation between events based on an ordering that may be present in the log file and an identifier that groups the security log traces together (e.g. internet address). Security event traces combine to create event sequences that aggregate to create a master graph model with statistical information on likelihood of

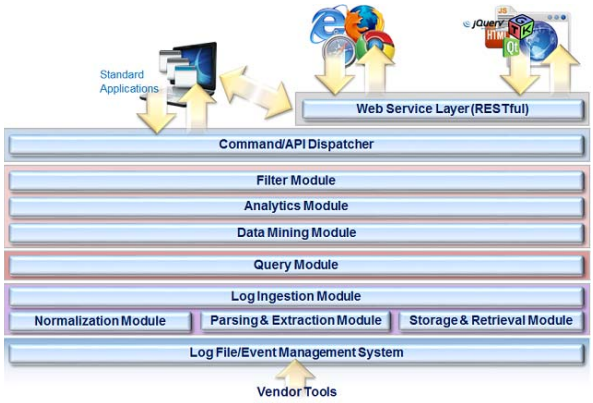

Fig. 3. LCA framework stack

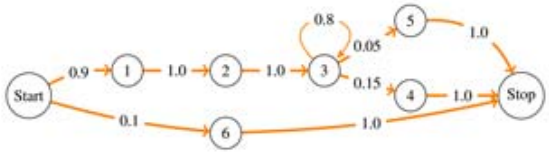

Fig. 4. Simplistic graph model

transitions between events, likelihood of the occurrence of events, and other relevant metadata.

As the number of event sequences increases, there is an increased need for memory to perform the mining process. To combat resource constraints of memory, event sequences are time bound in length such that the first and last event in a sequence cannot have a time difference greater than a specified threshold (typically 24 hours). To increase speed, the framework can also assume that trace entries are served in order (in some cases trace entries have been known to be out of order and delayed by days to weeks).

Each mapper ingests a portion of a file or subset of a group of files and learns a graph for that portion of the trace entries. As each mapper completes its task, its graph is merged with other graphs through a series of reducers to create a final master graph representative of all behaviors for a given slice of time.

Fig. 4. depicts a simplistic learned graph with edge transition likelihoods; however, in reality a graph model has many nodes with thousands of inter connecting transition links.

B. Anomaly Extraction and Ranking

Once the system learns a series of network agent behaviors as a graph model, we then discover anomalies within the network. We consider the master graph with likelihood transition information similar to a web graph with documents and thus apply PageRank over the model to discover the importance of any given event node with respect to others.

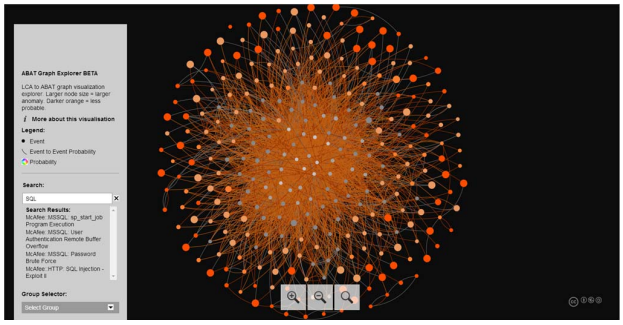

Fig. 5. Visualized graph model with events color coded according to how anomalous they are and sized according to their global probabilities.

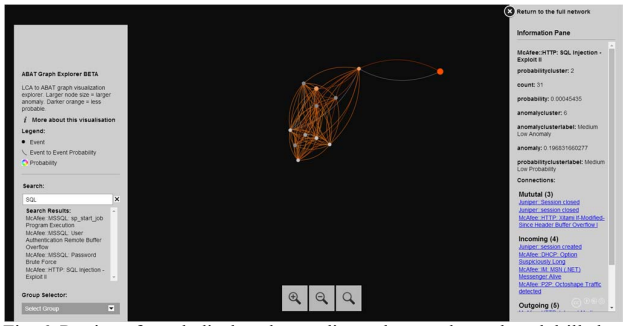

Fig. 6. Portion of graph displayed according to keyword search and drilled down into for additional details

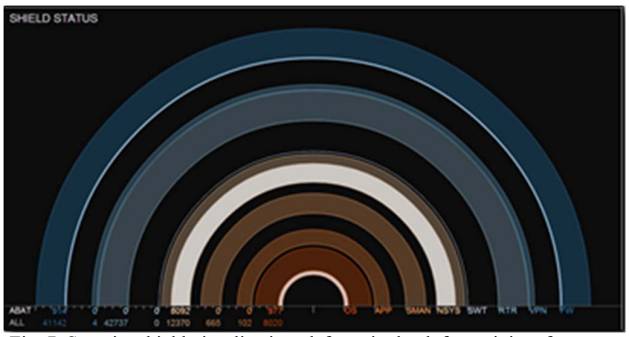

Fig. 7. Security shield visualization, defense in depth for activity of events at the firewall through to the OS level

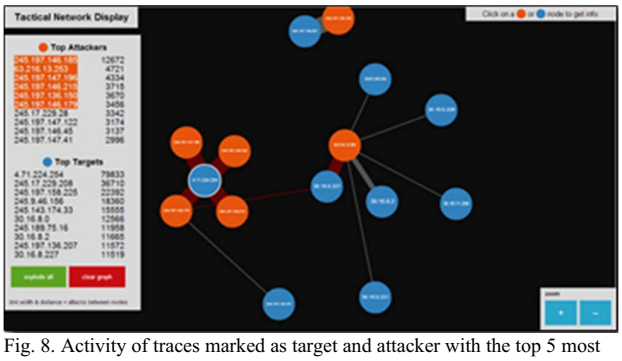

Fig. 8. Activity of traces marked as target and attacker with the top 5 most active anomalous target/attackers shown.

Opposite to the goals of search engines, which seek to return the set of most relevant nodes or documents in the graph, we seek to return the least relevant node events and hence most anomalous in this context. The system uses the set of probability values as the basis for automatically creating rules that contain the degree of anomalousness of streaming network event data.

Once anomaly probability values are calculated for every event node in a graph model we apply a k-means clustering with a randomly seeded centroid and a defined centroid displacement value indicating stability to rank the values into 5 distinct categorical groups: very-high, high, medium, low, and very-low.

VI. Deployment & Evaluation

A. Deployment

Our use case involved ingesting large amounts of network security events, detecting anomalous events in real-time based on an analytics model, storing the processed data in an in-memory database and creating custom, interactive visualizations to derive insights for the end user i.e. Chief Information Security Officer (CISO). The following sections describe the steps involved and the technologies used.

1) Data Discovery and Exploration

Our PoC dataset comprised of security events collected from the ArcSight SIEM tool from a Gobal 500 company. For model training purposes source data is encrypted, compressed, and obfuscated logs were stored on a staging server in 10-minute chunks of data of 4GB uncompressed. Each trace in the log file consists of 466 fields. Aggregate counts and data profiling was performed, while Tableau is used explore the data visually. Word clouds, bubble charts and aggregate views were utilized to understand and communicate the data profile to key stakeholders as well as inform the next step in the process.

The data set comprised of logs and trace entries from numerous devices on the network ranging from routers, switches, servers, etc. Each reporting system sent information to ArcSight, which graded events according to predetermined riskiness rules. All sensitive personally identifiable information (PII) was removed for our training purposes but relationships were maintained and reversible for the purposes of the security team on the ground.

2) Analytics

We used our LCA framework to identify events, event probabilities and statistics, and discover the temporal relationships between events. We applied the LCA framework to 3 months (3 petabyte) of security data to generate graphs with nodes representing the events, edges connecting events that are related to each other, the size representing the anomalousness and the color representing the probability of occurrence of the events (see Fig. 5.). Additionally, our framework provided the capability for analysts to drill down into the learned models to explore preceding and succeeding events for any given event (see Fig. 6.). Models were aggregated for a given hour for a given day to create a graph model. There are approximately 6.8 million traces, 25 GB of network trace data per hour, approximately 300 nodes per graph, and over 4000 edges per graph.

We used our Anomalous Behavior Analysis Tool (ABAT) module of the LCA framework to extract and categorize anomalousness scores for all events within a given model. These categorizations were fed into a real-time CEP engine as rules to

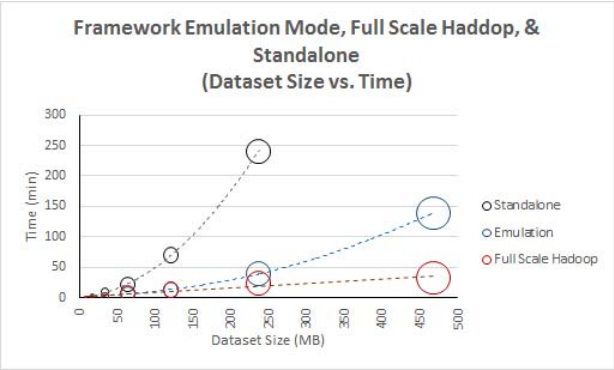

Fig. 9. Timing results of our framework modalities. Bubble size indicates graph complexity.

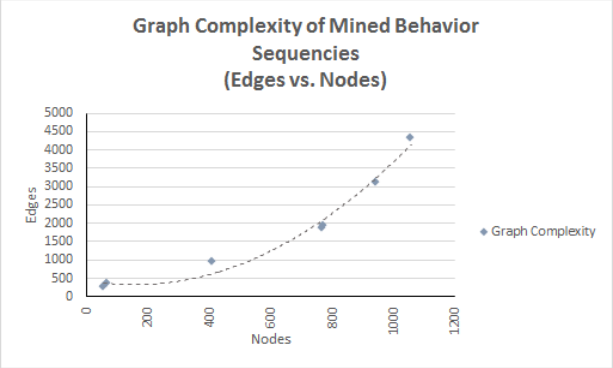

Fig. 10. Complexity of graph s produced.

grade new events for a given hour of a day to aid analysts and help provide context to risk assessments from ArcSight. For the PoC the LCA framework was built on Cloudera Hadoop (CDH5) running python Map-Reduce jobs. During the implementation of the system several anomalies were found that both surprised and alarmed security subject matter experts. For example, it was found that SQL injection exploits led to successful network logins. Additionally, in another case traffic that should be minimal, hence seen as anomalous, was in fact the opposite in the case of peer-to-peer network packets (particular peer-to-peer network applications are prohibited by company policy).

3) Real-time Processing

For real-time processing we used a data stream, a messaging queue (Kafka), a CEP engine (Apama) and an IMDB (GemFireXD). Each event trace from an input source is ingested into Kafka on a specific topic. Apama consumes the queue and performs rapid filtering, correlation, aggregation & detection of patterns across multiple data streams. Specifically we encode the Kafka consumer logic and the results of the ABAT automatically into Apama. When Apama encounters an anomalous event, it flags the events and stores it in GemFireXD.

4) Data Visualization

The goal of visualization is to make the data accessible to stakeholders and enable users and applications to connect to the data infrastructure in universally acceptable low latency ways. We created visualizations using D3.js, a JavaScript library for building custom visualizations. The first is a streaming visualization that provides the CISO and security incident response teams with board visibility into the most anomalous events compared to all events occurring in different security layers and depicting defense in depth (Fig. 7.). The teams can watch for fluctuations in the shield to ascertain volume of events at different levels.

The second visualization (Fig. 8.) enables visibility and exploration into the most anomalous behaviors of systems on the network through a force-directed graph showing the attacker and the target. This allows users to explore the attack patterns and proactively identify outliers to prevent event escalation.

B. Evaluation

We evaluated our system in two ways. First, we chose to demonstrate the capability of the graph mining of the system through timing tests. Secondly, we measured the precision and recall of the anomaly detection mechanism compared against categorization from subject matter experts (SME) accounting for accuracy and inter-rater agreement. For the second set of tests we also performed an ablation study showing system performance using only global likelihood estimators of events as a prediction of anomalousness. Precision and recall is calculated on the ablated system and full system.

1) Log Content Analytics Framework

To evaluate our analytics framework we chose to perform execution time tests in three modalities: standalone (executing timing tests on a single machine, single CPU, single core, 2GB RAM), emulation (parallel mappers on a single node cluster, 4 GB RAM), and full distribution (executing on a cluster of 4 machines, up to 2 mappers per node, 4 GB RAM). Evaluations were completed against dataset sizes ranging from logs of size 8 MB to approximately 470 MB in size and were a subset of a larger security incidence logs from the Global 500 Company as logged by the ArcSight SIEM tool. To compensate for system noise 10-fold evaluations were performed with results shown in Fig. 9.

As expected, the standalone and emulation versions demonstrated exponential growth and underperformed compared to the full distribution framework (the standalone was unable to complete all evaluations). The standalone and the emulation modes experience bursts in performance due to system constraints. The full distribution mode performed actions across several nodes from 2 to 8 mappers depending on data set size. The increase in mappers explain the linear behavior of the time. Results show that the size of the data is not the limiting factor but rather the complexity of the learned model. Fig. 10. depicts the complexity growth of the graphs as more information is ingested and mined.

2) Anomaly Detection and Categorization Evaluation

To measure relevancy of anomalous rankings, we sought SME input to grade the algorithm categorizations with respect to how the client network is actually populated with events. The purpose of which is to evaluate performance of correct system (precision) while maintaining event sensitivity (recall). This was broken into two parts with an evaluation of the full system and an ablation study using only global event probabilities as anomaly rankings. The ablation study demonstrates how the systems performs without the additional of linkage information

from a graph and uses simple probabilities of event occurrence in determining anomalousness.

a) Full Evaluation

For the full evaluation, each SME graded each anomaly category and determined whether a given event was properly categorized into one of five groups of anomalies: very high, high, medium, low, and very low. We quantified the comparison of results using the precision and recall as defined in equations (1) and (2) respectively (where tp refers to true positive, fp refers to false positive, and fn refers to false negative).

$$Precision = \frac{tp}{tp + fp} \quad (1)$$

$$Recall = \frac{tp}{tp + fn} \quad (2)$$

The precision and recall tests were performed on an average model for the same time of day and day of week. Approximate average precision and recall respectively for the categories were very high [0.7, 0.9], high [0.75, 0.7], medium [0.9, 0.9], low [0.8, 0.95], and very low [1.0, 0.4]. Using a Kappa test accounting for agreement by chance between raters, there was an 82% agreement and therefore the SME judgements align well with each other. SME metrics generally show good performance and high degree of agreement; however, categorizing events highlights difficulties with noisy signals and mismatched SME expectations of true anomalies occurring within the network.

b) Ablation Study

For the ablation evaluation, each SME graded each event according to how anomalous it is using only global probability of that event. This method of evaluation version provides a performance metric of the system without the anomaly probability ranking and anomaly categorization methodology. Events probabilities where classified into the same five categories used for the full system evaluation with the probability distribution consisting of very low probability, low probability, medium probability, high probability, and very high probability of occurrence (the five category boundaries were evenly placed across the probability distribution space). Event probabilities were inverted and mapped to the corresponding anomaly category (e.g. a very low probability of occurrence for a particular event correlates to that event being very highly anomalous) and resulted in the same five anomaly categories of: very high, high, medium, low, and very low. We quantified the comparison of results using the precision and recall as defined in equations (1) and (2). Similarly to the full system evaluation the precision and recall tests were performed on an averaged model. Approximated precision and recall respectively for the categories are very high [0.3, 0.6], high [0.1, 0.4], medium [0.2, 0.45], low [0.8, 0.35], very low [0.8, 0.14]. The reduced performance clearly demonstrates the poor quality of classifications of the ablated version of the system compared to the full system.

VII. Conclusions & Lessons Learned

Throughout the pilot, we learned the prevailing patterns of data acceleration with existing platforms tools and demonstrated a delivery of a robust, scalable, and contextually sensitive system. Some anomalies discovered surprised the SMEs and provided insight into the inner workings of the network that harkened investigations leading to successful mitigations. As we continue to display live demos and improve the deployed asset, we plan to implement more advanced real-time graph analysis techniques, automated architecture, and visualizations.

References

[1] A. Yelizarov and D. Gamayunov, "Adaptive Visualization Interface That Manages User's Cognitive Load Based on Interaction Characteristics", in Proc. VINCI, 2014, pp.1-1.

[2] *Fortscale,* Computer Software, Fortscale, San Francisco, Calif.

[3] J. Prewett, "Analyzing cluster log files using logsurfer," in Proc. of Annual Conference on Linux Clusters, 2003.

[4] J. Rouillard, "Real-time log file analysis using simple event correlator," in Proc. of Usenix LISA, 2004.

[5] *MetaGrid,* Computer Software, Red Lambda, Longwood, Calif.

[6] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina, "Web graph similarity for anomaly detection," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 19-30, 2010.

[7] *Palantir Metropolis*, Computer Software, Palantir Technologies, Palo Alto, Calif.

[8] R. Dijkman, M. Dumas, and L. Garcia Bannuelos, "Graph matching algorithms for business process model similarity search," *Business Process Management*, vol. 5701, pp. 48-63, 2009.

[9] S. Asanger and A. Hutchison, "Experiences and Challenges in Enhancing Security Information and Event Management Capability Using Unsupervised Anomaly Detection," in Proc. of ARES, 2013.

[10] *Splunk Enterprise*, Computer Software, Splunk, San Francisco, Calif.

[11] *Sqrrl Enterprise,* Computer Software, Sqrrl, Cambridge, Mass.

[12] W. Dickinson, D. Leon, and A. Podgurski, "Finding failures by cluster analysis of execution profiles," in Proc. of ICSE, 2001.

[13] W. Eberle and L. Holder, "Anomaly detection in data represented as graphs," *Intelligent Data Analysis*, vol. 11, no. 6, pp. 663-689, 2007.

[14] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Mining console logs for large-scale system problem detection," in Proc. of SysML, 2008.

[15] Y. Tian, R. McEachin, C. Santos, D. States, and J. Patel, "Saga: a sub-graph matching tool for biological graphs," *Bioinformatics*, vol. 23, no. 2, pp. 232-239, 2007.