Defense at Scale: Building a Central Nervous System for the SOC

Joseph Zadeh, George Apostolopoulos, Christos Tryfonas, Muddu Sudhakar
Splunk, Inc.

ABSTRACT

Data driven security is advocated as a way to augment traditional workflows in security operations. The goal of this data driven approach is to design a system that automatically labels escalation events as well as helping perform reproducible forensic tasks. Additionally the problem of tracking the evolving threat surface is attacked by leveraging a highly expressive composition of distributed systems called a Lambda Architecture.

1. INTRODUCTION

The approach stressed in this paper is a common sense application of best practices combined with optimized technological applications. One of the fundamental principles we advocate for accelerating defense at scale is the use of a Lambda Architecture [1], [2] to distribute the load of forensics/analytics jobs and to have a constantly running security oracle that gets better the more it is used by operators. The other major paradigm we advocate is a system engineering approach to mapping the threat surface to behavioral/classical IOC's and more importantly "quantifying" the tractable security uses cases in order to maximize the human/technological interaction.

How do you model an adversary with asymmetric access to resources and with tactics and techniques that apply to a continuously evolving threat surface? We answer this question by mixing analytics with traditional SOC technologies to achieve operational efficiencies in workflows and personnel utilization. A primary challenge faced in the design of such a system is engineering it to stay current with the evolving threat surface. Not only that but the early use of machine learning and the limitations first generation SIEMs brought on large security operations has compounded the frustration and skepticism surrounding nextgen solutions (for instance see [3], [4]). In a nutshell defense has matured at a slow pace when it comes to intrusion detection and continues to trail with the agility and sophistication of the current threat landscape.

One aspect of this asymmetry was pointed out by Mudge in his Blackhat keynote of 2011 [5]. In the keynote Mudge highlights some deep aspects of defense compared to offense by using several key metrics. One metric he notes involves lines of code comparing malware samples to defensive software (on average one malware sample was 160 lines of code whereas a defensive product like AV or FW was potentially millions of lines of code). A takeaway from this analysis is that lines of code are a proxy for work or more importantly money. This observation in terms of workflows means that for each unit of resources on offense it requires 10X-1000X more resources on the defensive side. How do we reconcile such an asymmetry especially given his additional observations on incentive structures and the complex threat surface inherent in modern software?

1.1 Central Nervous System

Enabling the SOC with tools that are agile and responsive to a rapidly changing threat starts with getting maximum value from every piece of evidence available in an enterprise. The streams of information that represent various logging sources are the core sensory input for a computational platform designed to help SOC personnel. Not only the logs but the behaviors of the users and devices provide key patterns with which to extract all sorts of rich security-centric information: user behavior can be used to automatically determine different types of devices on a network, build a statistical whitelist of most popular internal/external hosts etc. These type of behavioral patterns are unique to each enterprise and can in the cases of large networks represent very valuable telemetry data that should be prioritized and correlated with traditional intelligence signals.

Looking at the threat surface there are a large number of isolated use cases that we can reduce to tangible algorithms and workflows potentially if the use case is not already covered by point solutions. This roadmap approach of using custom analytics in collaboration with the existing signal from known technologies is the core idea we try to stress in this paper. From the authors perspective point solutions can in fact provide a stable source of signal for the complex attack surface. For example host information along with email logs, DNS logs and outbound web traffic can enable an active response workflow [6]..

Components to a successful Central Nervous System:

1. General data processing framework
2. Immutable append only logging

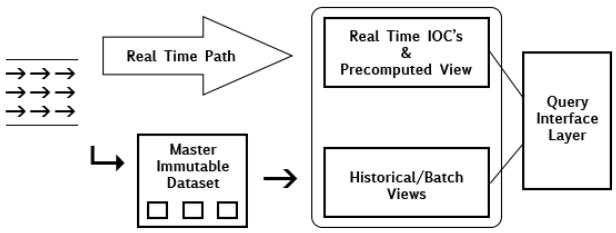


Figure 1: Lambda Architecture Example

3. Machine learning based ETL (Extract/Transform/Load)
4. Distributed crawlers
5. Automated identity/session resolution and fingerprinting
6. Formal evidence collection protocol for automated labeling of incident response data
7. Analytics Metrics and establishing benchmarks for intel/analytic data

The above list touches on a range of diverse and interdispinar technical topics. The focus of this whitepaper centers around the data processing framework design and why it is important to isolate parrlele paths of real time and historical computation to properly decouple specfic bottlenecks in the paths of security data. We have seen limitations from a SOC standpoint when defending against complex sets of behaviors up and down the "Pyramid of Pain" [7] The focus on research to mitigate such challenges in defense has prompted DARPA for instance to create programs such as ADAMS [8]. A properly designed security central nervous system can function as a key tool in scaling technology to remain flexible in the face of evolving threats.

2. LAMBDA ARCHITECTURE

This section gives a brief overview of the Lambda Architecture and why it is such a useful methodology for intrusion detection. The term was originally coined by Nathan Marz to describe a generic problem solving approach that scales to extremely large data processing tasks. For a more comprehensive picture of the principles outlined in this section we recommend the excellent book "Big Data: Principles and best practices of scalable real-time data systems [1]" by Nathan Marz and James Warren. During remarks on how Twitter leveraged this type of platform at the 2011 Cassandra conference Nathan observed why such a system is valuable [9]:

> "We have our batch layer constantly override the real time layer so that if anything ever goes wrong its automatically corrected within a couple hours. That batch/real time approach is really general. It works for pretty much any problem. Its something I really recommend its really a deep topic but that is the approach we use . . . "

What the quote does not capture quite well is the ideas behind how traditional databases and similar technology have been baked on a set of standards that mix the querying of data with the storage of data. This mixture leads to all sorts of complexity and scale issues and can be avoided to a large extent by abstracting the different components of transactional databases such as Read, Write, Update Delete and managing them over different layers of an overall architecture. A properly designed immutable and distributed system with the right paths for different computations such as batch and real time can optimize common security workflows (see figure 2).

2.1 Building Blocks

The overall architecture is broken down into three layers: real-time, batch and a serving layer. The batch layer can be though of as maintaining an immutable master copy of all the original data (possibly over a distributed file system). The main principles that are advocated in the Lambda Architecture can be expressed in three simple equations that represent the break down of the different layer operations [1], [10]:

- batch view = function(all data)
- realtime view = function(realtime view, new data)
- query = function(batch view, realtime view)

In terms of the open source community the batch layer can be built using Hadoop [11] or Spark [12] for instance and the realtime view has Apache Storm [13]. From a commercial perspective specialized vendors like Splunk offer the capability of maintaining a scalable distributed Lambda architecture without the complexity of designing the system from scratch.

2.2 Immutable Logging

The keys to a lambda architecture start with a data model that is append-only and immutable. This means your master data set has certain beneficial optimization properties compared to a traditional transactional database for instance. A great practical reason why immutability is important for to security logging can be found in [5] by Peter (Mudge) Zatko:

> "A lot of folks don't realize this and I try to point it out when people are doing defense. Just having good logging in a separate system that is immutable. So the equivalent of a line printer in

your data center that is spewing out syslog stuff is a big deterrent - a write only media device. For many nation states or industrial intelligence communities for industrial espionage or clandestine activities you would rather fail at your mission and go unnoticed than actually succeed and draw attention to yourself."

2.3 In Defense of Defense

Machine Learning as a toolset in our problem space has significant limitations. Adversarial drift implies algorithms will constantly have to adapt [14]. Furthermore designing intelligent software will be limited by the sub-problems that admit tractable solutions via machine learning. Thats why on modern hardware at best we can use tools from artificial intelligence and expert systems to augment manual security workflows.

Accuracy is synonymous with consistency of labeled data as well as consistency of the overall digital evidence collection processes. In cybersecurity the issue with malicious attacks is that they are often times new patterns (0-days) that have never been observed before in the data and such new behaviors mean any supervised learning algorithm will most likely misclassify the attack behavior. The lack of labels has a theoretical explanation for why the algorithms suffer called "No Free Lunch Theorems" [15], [16]. With that in mind a catch all mechanism should be a first priority when it comes to building perimeter open source technologies.

The issue to isolate here is the importance of baking in fall throughs naturally into all defensive systems that security teams build. Also and possibly more importantly finding a normalized way to store incident response data to use to benchmark vendor solutions who claim they can catch certain behaviors and to build internal systems if desired. Spending effort on establishing the proper workflow for evidence collection and labelling will pay dividends in the future thanks to current trends in automated feature learning and deep architectures for classification. For instance see [18], [19] and [20]. Combining the power of cutting edge applications in automated feature learning with a security minded workflow like outlined in [21] is a great example of blending the best of human performance with algorithms.

Particularly from an automation standpoint we can use some of the latest neural network research to design an active batch and real time workflow. We can back-propagate the learning process dervied from security investigations automatically using a deep neural networks to extract meaningful patterns across SOC evidence collection results. These features then can automatically be fed into an updated behavioral IOC layer that will deploy the latest patterns in a sandbox and be tested against benchmark data that has been manually labeled before being reviewed for production.

2.4 Reactive Defense: Commodity Crawlers

The main premise here is that with distributed architectures we can not only use a multi-threaded parallel computational engine to constantly be collecting, cataloging and correlating evidence but we can also spawn worker processes for specific forensic tasks. Two key augmentations in this regard are distributed crawlers for analyzing both internal/external data. For example watering hole detection, custom forensic crawling of internal systems and automating key forensic tasks that require scraping intel repos and related evidence collection [22].

Figure 2: Model Value vs. Cost of Validation

3. SECURITY ANALYTICS ROI

How do you rank order the value of a cybersecurity analytic? How do the models we build add value to a customers existing operation and point solution coverage? Due to the fact that the threat surface changes rapidly it is important to create expressive analytics to capture key properties of patterns that are useful for forensics and similar security centric processes at the same not overlapping with pre-existing technologies.

In the distributed instance this modeling approach becomes extremely important because we might have individual analytics distributed across a multi node-cluster. Such parallelism comes with limitations though especially in the streaming path and that is why it is so important to map use cases to optimized strategies leveraging individual components in the Lambda architecture. We advocate a way to quantify and assess the value of blending behavior based analytics with point solution and existing defensive technologies. We start with a simple breakdown of the motivations for our strategy:

1. Maximizing personnel efficiency can be achieved by first breaking down the threat surface into actionable use cases

2. Once top use cases are identified ranking them by ROI for automation with ML/Data driven security will give

a roadmap for automating SOC workflows and mapping automated workflows to the the threat surface

3.1 Analytic Value

We seek a way to compare security analytics via a simple metric that is constantly updated by natural SOC workflows. A simple example helps illustrate our purposes. Our total value for two different models can be compared as follows. Lets say we have built an analytic to detect anomalies in VPN logins which we find on average takes our internal security team 5 hours of investigation and produces 20 individual events on average to investigate per week.

Assume we also have custom active directory analytic feeding into our SOC based that looks for anomalies in AD tree structure and macro properties of the social network in LDAP data but the output of this model takes on average 20 hours of investigation per alarm and produces 15 events on average a week. One final aspect of this ROI exercise is we need to take into account the impact risk or the cost to the organization if the particular attack vector goes undiscovered as well other more complex variables.

$$\$ \text{ Model Value} = \$ \text{ Impact Risk} - (\$ \text{ Development} + \$ \text{ Validation})$$

https://www.youtube.com/watch?v=Rr8-SPS62js An example of a way to use this type of intuition to rank the value of models we described above (plus many more) can be found in 2. The main takeaway for the SOC from the above discussion is to consider the cost of validation for all security solutions when compared to how much risk is mitigated. When the SOC is faced with the additional need to rank order multiple analytics solutions a simple breakdown of value versus "security cost" is a great way to help roadmap an analytics priority list and find sweet spots for best return on investment. Be warned assigning dollar cost to future cyber security events at best is a dark art on its own but in combination with other more ephemeral variables should be thought of as a way to motivate a more broader search for quantifying and describing value among cybersecurity analytics.

4. CYBERSECURITY ANALYTICS

In this section we try to illustrate hands on examples of analytic models and how they are best realized across the multiple layers of the Lambda architecture. The word analytic can mean a lot of different things depending on context and industry. In the authors experience we use it to describe a set of algorithms in combination with the deployment and operationalization of such algorithms through a distributed architecture. We focus primarily on a couple of key problem areas in cybersecurity analytics:

1. Combining point solutions data and IOC's with behavioral signals in a workflow that automatically correlates intel in real time with historical calculations
2. Theoretical example involving random walks to model user behavior

4.1 Blending Traditional IOC's with Behavioral IOC's

Part of the power of using an immutable and distributed architecture is we can process large amounts of heterogenous information and correlate to pre-commuted security views in real time. A first application of blending such information is developing a scoring routine for isolating strange traffic patterns and providing as much automated attribution and risk classification as possible. To model the situation we run two processes in parallel:

1. Periodically we run a batch graph cut/label propagation step.
2. Simultaneously we run a real time scoring workflow for consuming the output of process 1 along with new IOC's generated from points solutions. To model risk for a heterogeneous set of features we build a scoring function g to automatically determine risk for a given input cluster of nodes

Simple examples of such a scoring function g are heuristics, bayesian scoring methods and more exotic forms of believe fusion such as Dempster-Schafer theory [23], [24], [25]. For completeness we mention specific implementations designed to blend evidence given security context like [26] and [27]. For our example we will assume a simple rule for fusing distinct pieces of evidence based off Naive Bayes model (an illustration of such evidence fusion is found in figure 4).

The use of graph based techniques at resolutions of the file system level all the way up to the social graph structure of an entire corporate intranet lend themselves very well to batch layer computations for example see [28], [29], [30]. By segmenting different computations to different paths we can isolate complexity and stay agile in terms of providing up to date intelligence for behaviors that are naturally sequential in nature and best computed in the real time path (for example exploit chain detection is a behavior that is tractable in a real time path because it is highly sequential in nature [31]).

Real time streams of IOC's can also be included in the path for example AV alarms from a host, critical alerts from Palo Altos or Proxy devices, FireEye/Sandbox hits etc. Once these alarms are fed continuously into our system we can begin running batch graph analytics [32] in combination with a lightweight scoring method (implementation of g) to combine the distinct point solution signals with any additional behavioral characteristics that are local or global to a specific graph segment.

Finally we list one possible implementation for modeling behavioral IOC's based off standard the Bayes rule:

Behavioral Indicator of Compromise (IOC) *Let W be any sequential data representing for instance an incoming stream of bytes or content types from proxy logs. For example let $W_1 =$ text/html,*

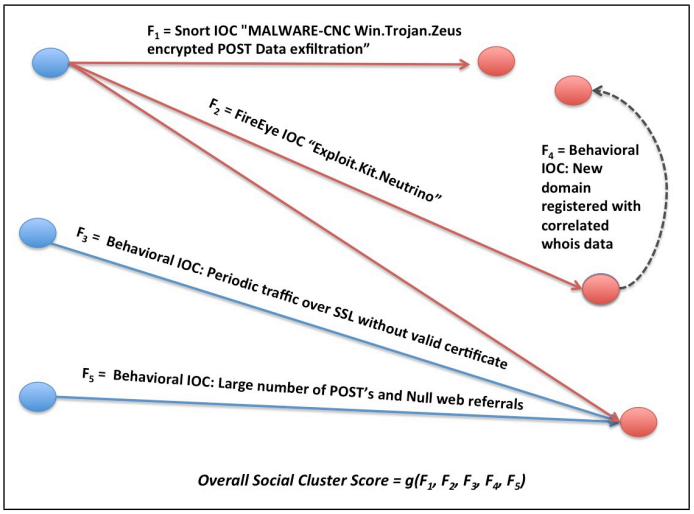


Figure 3: Example of calculating a risk score for a cluster of traffic where g is a heuristic or a function classifier trained via a supervised learning algorithm

$W_2 = application/java\text{-}archive, W_3 = application/octet\text{-}stream$. *Apply the Bayes "chain rule" to get*

$$\mathbb{P}(W_n|W_{n-1},\ldots,W_1) = \frac{\mathbb{P}(W_1,\ldots,W_n)}{\mathbb{P}(W_1)\mathbb{P}(W_2|W_1)\mathbb{P}(W_3|W_2,W_1)\ldots\mathbb{P}(W_{n-1}|W_{n-2},\ldots,W_1)}$$

Rearranging formula 1.1 *gives a method for approximating the probability of the sequence occurring:*

$$\mathbb{P}(W_1,\ldots,W_n) = \mathbb{P}(W_1)\prod_{i=2}^{n}\mathbb{P}(W_i|W_{i-1},\ldots,W_1)$$

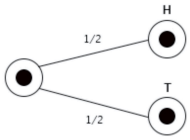
We then train a model on the probability of each of the sequences of events occurring in order. For instance if we are creating a model for exploit chain detection we can use as training input all the content sequences of small length occurring in outbound proxy data for the last year.

4.2 Modeling Users as Processes

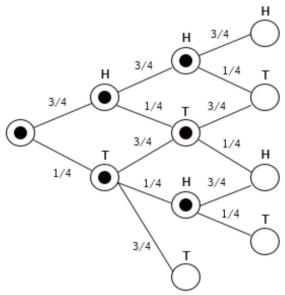
This section is intended to show some more advanced examples of ways to capture specific behaviors from both a real time and batch perspective. We propose a simple prototype for modeling individuals via a probabilistic process called Fractional Brownian motion.

4.2.1 Fractional User Behavior

The example we develop focuses on a specific analytic tracking user behaviors through outbound Layer-4 and Layer-7 data. We will rely on this model for approximating a single entities digital footprint by decomposing individual user (or IP or MAC etc..) patterns into a composition of processes. Formally what we mean by a process is described below:



(a) Hurst Exponent = $\frac{1}{2}$



(b) Hurst Exponent > $\frac{1}{2}$

Figure 4: Approximate probabilities for random walks represented by coin flips $\Omega = \{T, H\}$

Defintion 1 *A (stochastic) process $S = \{S(t) : t \geq 0\}$ is a collection of random values evolving over time. Furthermore the values of S are not only random but they can occur in a high dimensional space D. In formal notation we write $S : [0, T] \times \Omega \to D$ where D is some arbitrary data set (or meta-data) and Ω can be thought of as the sample space.*

The literature on processes has a rich history from finance to biology to physics. The process important to our model is called Brownian motion and it has been studied for over two centuries by biologists, statisticians and even Einstein [33]. Brownian motion is the canonical example one will arrive at if we need a to approximate a random walk on a computer with the following theoretical properties:

1. The previous history only depends on the last step (the transition is Markov)
2. The path of the random walk is continuous and fractal (the process is self-similar)

More formally we say Brownian motion is the continuous time generalization of a random walk that is Self-Similar and Markovian. In addition the process can be seen as a stationary time series with independent and identically distributed Increments [34]. To further complicate things we will use a generalization of the above definition called Fractional Brownian motion (fBm). This more complex process was first studied by the hydrologist Hust [35] as well as Mandelbrot [36]. For a more comprehensive breakdown of fBm see Chapter 5. of [37].

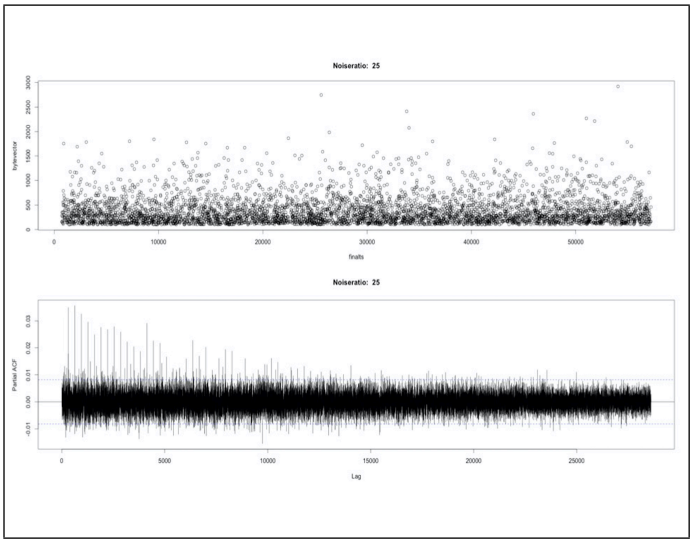


Figure 5: Single Source Traffic with Periodic C2 Component

The theoretical aspects of Brownian motion are still an active area of research but thankfully there are good open source implementations available that abstract away many of the details. We recommend for instance the R Library fArma [38] for simulation and experiments of fBm along with complex related time series. The main property of fBm we will exploit in our model of user behavior is that the process exposes a single parameter (called the Hurst parameter H) to represent the history of the process. This is very useful in streaming computations if one seeks to find an alternative to computing transition probabilities in a multi stage Markov process but only if the process is self-similar. To see how H comes into the picture we list one more definition:

Defintion 2 *Fractional Brownian motion*
A Fractional Brownian motion with Hurst parameter $H \in (0, 1)$ *is a centered Gaussian process* $B = \{B(t) : t \geq 0\}$ *with covariance given by* $\texttt{Cov}(\texttt{t}, \texttt{s}) = \frac{1}{2}(|\texttt{t}|^{2\texttt{H}} + |\texttt{s}|^{2\texttt{H}} - |\texttt{t} - \texttt{s}|^{2\texttt{H}})$.

The main benefit of this formulation of fBm is that computationally we can maintain a single parameter H to study the historical behavior of a sessions traffic pattern. We can offload complex parts of the computation for updating statistics of H to batch layers and use the real time layer to do threshold detection and lightweight approximations given new data about a session. From a historical perspective it is important to note that a classic paper [39] in network theory shows ethernet traffic is self similar which is a basic requirement to doing certain types of anomaly detection with H but in general it is not clear that all continuous valued processes driven by user traffic are self-similar.

4.2.2 *Streaming Command and Control Detection with Hurst Exponents*

Let U represent a single users traffic and E a unique external domain or IP. Then a model of user behavior over time is given by assigning to each session (U, E_i) a process S_i which we assume is fractional Brownian motion. Furthermore we restrict the model to the case when input logs are from proxies, firewalls or similar perimeter appliances. Given a user U our model of the log behavior is given as $U = (S_1, S_2, \ldots, S_N)$ where each S_i is a fractional Brownian motion possibly representing only a single event or even continuous stream of constant traffic.

When we build all sessions into a single model we get a picture of the population that evolves with the individual users micro behaviors (see figure 6). This representation is both useful because it parallelizes as well as scales to large population of users which enables data driven detection of macro economic trends in the larger population. For a single external domain we can read in all session Hurst parameters and explore both supervised and unsupervised approaches to detecting command and control and other risky traffic patterns.

Algorithm 1 Anomaly Detection on $\mathbf{H}$

```
while AnomalyCheck(H(t_k))! = true do
  if k = 0 then
    H(t_0) ← hurstBatch(H)
  else
    H(t_{k+1}) ← hurstRealTime(H(t_k))
    H ← (H, H(t_{k+t}))
  end if
  if Batch Job == true then
    (lofAnomalyCheck||factorAnomalyCheck)
    H ← hurstBatch(H)
  else
    heurisitcAnomalyCheck
  end if
end while
```

Fix a session S with a vector of Hurst parameters called $\mathbf{H} = (H_0(t_0), H_1(t_1), \ldots, H_k(t_k), \ldots)$. We want to describe a general procedure for updating $\mathbf{H}$ as well as checking for a state change that works in an embarrassingly parallel fashion. In particular each S can be isolated in a single thread of memory and therefore sharing state across sessions will best be achieved in a batch or pre-computed view. The key algorithmic task we want to develop is a sampling based approach to determining the next update to the memory parameter for a single session and when to raise an alarm if a session is determined to be command and control.

The vector of parameters $\mathbf{H}$ will be the main object we perform anomaly detection on and can be thought of as a lightweight representation of a single sessions history. For our pseudo implementation we expose methods for computing anomalies on the vector of Hurst parameters and the typical batch and real time worker processes. The first method we implement is called `heuristicAnomalyCheck` and is designed as a anomaly detection heuristic based off expert

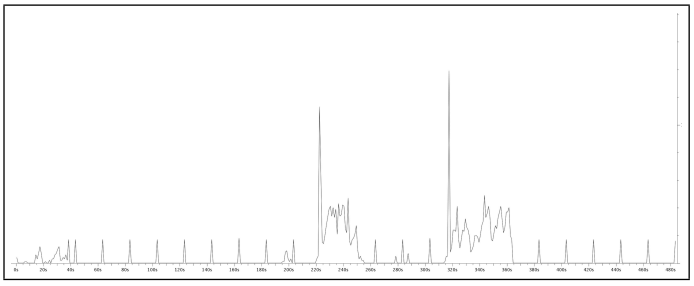


Figure 6: RAT C2 Traffic and Heartbeats [44].

input. For a real time anomaly check we refer to `hurstRealtime` as an approximation to the rescaled range analysis in [42] and for a batch layer computation we will refer to `hurstBatch` as the algorithm outlined in [43]. The second historical method we implement is a basic local outlier factor detection [40] on the space of all hurst vectors for a single data set: `lofAnomalyCheck`. Finally the last method that is important is called `factorAnomalyCheck` where this is an example of a supervising learning technique and will only be useful if we have an abundant sample of labeled data (the particular learning methods we leverage are known as factorization machines [41]). The core workflow is highlighted in Algorithm 1.

5. CONCLUSION

We have described a way to use scalable data principles in combination with prioritization and mapping of use cases to analytic problems but this was only an outline for part of the solution needed in the larger problem of cybersecurity defense. One important technical topic that was omitted from this paper was about model forgetting, re-training and general aging within the entire security workflow and there are a multitude of other such issues that face a large scale defensive data driven operation.

Whats more, the cost large scale operations have maintaining twenty four hour defensive programs in the face of constantly evolving threat tactics means R&D has to step up and deliver solutions that ease such a massive burden on human personnel. It is our position that with the use of the right modeling philosophy in combination with highly adaptable data systems opportunity exists to build a next generation security central nervous system that can help balance the asymmetric nature of offense versus defense.

6. ADDITIONAL AUTHORS

7. REFERENCES

[1] Nathan Marz and James Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications, 2013.

[2] Nathan Marz. How to beat the CAP theorem. nathanmarz.com/blog/how-to-beat-the-cap-theorem.html, December 30 2011.

[3] Anton Chuvakin. 9 Reasons Why Building a Big Data Security Analytics Tool is Like Building a Flying Car. blogs.gartner.com/anton-chuvakin/2013/04/15/9-reasons-why-building-a-big-data-security-analytics-tool-is-like-building-a-flying-car/, April 15 2013.

[4] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316, May 2010.

[5] Peter (Mudge) Zatko. Black Hat USA: How a Hacker Has Helped Influence the Government and Vice Versa. youtu.be/kZk9fsQisI8, 2011.

[6] Monzy Merza. Active response: Automated risk reduction or manual action? www.rsaconference.com/events/us15/agenda/sessions/2012/active-response-automated-risk-reduction-or-manual, 2015.

[7] David Bianco. The Pyramid of Pain. detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html, January 17, 2014.

[8] DARPA. Anomaly detection at multiple scales. opencatalog.darpa.mil/ADAMS.html, 2014.

[9] Nathan Marz. Cassandra NYC 2011: The storm and cassandra realtime computation stack. youtu.be/cF8a_FZwULI, December 30 2011.

[10] Nathan Marz. Apache storm. youtu.be/ucHjyb6jv08, 2013.

[11] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2009.

[12] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[13] Nathan Marz. Apache storm. storm.apache.org, 2014.

[14] Marius Kloft and Pavel Laskov. Online anomaly detection under adversarial impact. In *I*, volume 9 of *JMLR Proceedings*, pages 405–412. JMLR.org, 2010.

[15] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *Trans. Evol. Comp*, 1(1):67–82, April 1997.

[16] David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390, October 1996.

[17] Michael I Jordan. Ama: Michael i jordan. www.reddit.com/r/MachineLearning/comments/2fxi6v/ama_michael_i_jordan, 2014.

[18] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[19] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. Technical Report Arxiv report 1206.5533, Université de Montréal, 2012.

[20] Eric J. Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martins, and Meinard Müller, editors, *ISMIR*, pages 403–408. FEUP Edições, 2012.

[21] Jack W. Stokes, John C. Platt, Joseph Kravis, and Michael Shilman. ALADIN: Active Learning of Anomalies to Detect Intrusions. Technical Report MSR-TR-2008-24, Microsoft, March 2008.

[22] G. Rush, D.R. Tauritz, and A.D. Kent. Dcafe: A distributed cyber security automation framework for experiments. In *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pages 134–139, July 2014.

[23] Jurg. Kohlas and Paul-Andre Monney. Theory of evidence: A survey of its mathematical foundations, applications and computational aspects. *Zeitschrift Operations Research*, 39(1):35–68, 1994.

[24] Audun Josang, Javier Diaz, and Maria Rifqi. Cumulative and averaging fusion of beliefs. *Information Fusion*, 11(2):192 – 200, 2010.

[25] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Ann. Math. Statist.*, 38(2):325–339, 04

1967.

[26] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *ACM CoNEXT '10*, Philadelphia, PA, December 2010.

[27] Hoda Eldardiry et. al. Multi-source fusion for anomaly detection: using across-domain and across-time peer-group consistency checks. *JoWUA*, 5(2):39–58, 2014.

[28] Pamela Bhattacharya, Marios Iliofotou, Iulian Neamtiu, and Michalis Faloutsos. Graph-based analysis and prediction for software evolution. In *International Conference on Software Engineering (ICSE)*, Zurich, Switzerland, June 2012.

[29] Yi Yang, Marios Iliofotou, Michalis Faloutsos, and Bin Wu. Analyzing interaction communication networks in enterprises and identifying hierarchies. In *IEEE International Workshop on Network Science (NSW)*, June 2011.

[30] Brian Gallagher, Marios Iliofotou, Tina Eliassi-Rad, and Michalis Faloutsos. Homophily in application layer and its usage in traffic classification. In *IEEE INFOCOM*, San Diego, CA, USA, March 2010.

[31] Hesham Mekky, Ruben Torres, Zhi-Li Zhang, Sabyasachi Saha, and Antonio Nucci. Detecting malicious HTTP redirections using trees of user browsing activity. In Giuseppe Bianchi, Yuguang M. Fang, and Xuemin S. Shen, editors, *INFOCOM 2014, 33rd IEEE International Conference on Computer Communications*, pages 1159–1167, Los Alamitos, CA, USA, April 2014. IEEE.

[32] Damien Faya Hamed Haddadibi, Steve Uhligc, Liam Kilmartind, Andrew W. Mooreb Jerome Kunegise, and Marios Iliofotouf. Discriminating graphs through spectral projections. *Computer Networks*, 55:3458–3468, October 2011.

[33] A. Einstein. Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der Physik*, 322:549–560, 1905.

[34] Patrick Billingsley. *Probability and Measure*. Wiley-Interscience, April 1995.

[35] H. Hurst. Long term storage capacity of reservoirs. *Transaction of the American society of civil engineer*, 116:770–799, 1951.

[36] B. B. Mandelbrot and J. W. van Ness. Fractional Brownian motions, fractional noises and applications. *SIAM Review*, 10:422–437, 1968.

[37] David Nualart. *The Malliavin calculus and related topics*. Probability and its applications. Springer, Berlin, Heidelberg, New-York, 2006.

[38] Diethelm Wuertz et. al. farma: ARMA time series modeling. cran.r-project.org/web/packages/fArma/index.html, June 24 2013.

[39] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Netw.*, 2(1):1–15, February 1994.

[40] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[41] Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[42] Benoît Mandelbrot and J. R. Wallis. Robustness of the rescaled range R/S in the measurement of noncyclic long run statisticaldependence. 5:967–988, 1969.

[43] Alexandra Chronopoulou and Frederi G. Viens. Hurst index estimation for self-similar processes with long-memory. In *Recent development in stochastic dynamics and stochastic analysis. Dedicated to Zhi-Yuan Zhang on the occasion of his 75th birthday.*, pages 91–117. Hackensack, NJ: World Scientific, 2010.

[44] Mila Parkour. Contagio malware dump. contagiodump.blogspot.com/2013/04/collection-of-pcap-files-from-malware.html, 2015.