AFRL-RY-WP-TR-2019-0140

A DIAGNOSTICS APPROACH FOR PERSISTENT THREAT DETECTION (ADAPT)

Ryan Wright, Alan Fern, Anthony Williams, James Cheney, Ghita Berrada, and Sid Ahmed Benabderrahmane

Galois, Inc.

NOVEMBER 2019
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the USAF 88th Air Base Wing (88 ABW) Public Affairs Office (PAO) and is available to the general public, including foreign nationals.

Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RY-WP-TR-2019-0140 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

* //Signature/

DAVID A. KAPP
Program Manager
Resilient and Agile Avionics Branch
Spectrum Warfare Division

//Signature/

DAVID G. HAGSTROM
Chief
Resilient and Agile Avionics Branch
Spectrum Warfare Division

//Signature/

JOHN F. CARR, DR-04
Chief, Spectrum Warfare Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show "//Signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
November 2019	Final	26 June 2015 – 30 June 2019

4. TITLE AND SUBTITLE
A DIAGNOSTICS APPROACH FOR PERSISTENT THREAT DETECTION (ADAPT)

5a. CONTRACT NUMBER
FA8650-15-C-7557

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)
Ryan Wright, Alan Fern, Anthony Williams, James Cheney, Ghita Berrada, and Sid Ahmed Benabderrahmane

5d. PROJECT NUMBER
1000

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
Y1AX

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Galois, Inc.
421 SW 6th Ave., Suite 300
Portland, OR 97204

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

Defense Advanced Research Projects Agency
DARPA/I2O
675 North Randolph St.
Arlington, VA 22203

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYWA

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2019-0140

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
PAO Case Number: 8ABW-2019-5140, cleared 23 October 2019. This work was funded in whole or in part by Department of the Air Force Contract FA8650-15-C-7557. The U.S. Government has for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U. S. Government. Document contains color.

14. ABSTRACT
Over the course of the Transparent Computing program, the ADAPT team developed a system for Automated Detection of Advanced Persistent Threats (APTs). The core developments of the Quine distributed graph database together with a newly developed technique for categorical anomaly detection provided the capability to perform probabilistic analysis for all system activity at arbitrarily high speeds. Using policies defined once by a team of experts, the system is able to effectively find and describe considerable APT activity and produce meaningful summaries for a human analyst.

15. SUBJECT TERMS
anomaly detection, advanced persistent threat, APT, transparent computing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			David A. Kapp
Unclassified	Unclassified	Unclassified	SAR	54	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

TABLE OF CONTENTS

Section **Page**

List of Figures

Figure **Page**

List of Tables

Table **Page**

1.0 SUMMARY

Over the course of the Transparent Computing program, the ADAPT team developed a system for Automated Detection of Advanced Persistent Threats (APTs). The core developments of the Quine distributed graph database together with a newly developed technique for categorical anomaly detection provided the capability to perform probabilistic analysis for all system activity at arbitrarily high speeds. Using policies defined once by a team of experts, the system is able to effectively find and describe considerable APT activity and produce meaningful summaries for a human analyst.

2.0 INTRODUCTION

If an advanced attacker carefully exploits a system, there is no way to know. The ADAPT project has been researching new capabilities to change that situation.

Modern computers are very fast and rely on an astounding level of carefully orchestrated complexity to perform correctly. Creating a system to perform the *desired* behavior correctly is often very difficult. Making that system perform *only* the desired behavior is nearly impossible. Skilled and well-funded adversaries often need to find (or buy) only a single exploit to gain entry into a computer system. Once in, the attacker can move about largely unnoticed.

The most sophisticated attackers are careful in their exploration, exploitation, and exfiltration of a compromised computer system. Advanced attackers will work “low and slow” making small careful steps as they progress on their malicious mission. These careful steps leave no trace currently detectable by operating systems, anti-virus, or other security programs. And the stakes are large.

The DARPA Transparent Computing program begins by admitting how dire the current situation is, and has directed the resources of many incredible researches to advancing the state of the art and creating means by which we might detect these Advanced Persistent Threats (APTs). Teams on this program have created tools to observe the behavior being done by a running computer system, and analyze that behavior to pinpoint the APT and explain what they have been doing.

The ADAPT team has been working on the analysis side of this problem. The goal of ADAPT has been to create new methods for the Automated Detection of Advanced Persistent Threats. We have succeeded in this goal by creating a scalable, high-throughput management system for the graph data representing system traces from any number of instrumented machines, and the development of a new class of anomaly detection algorithms: Categorical Anomaly Detection.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

As a practical strategy to unify the goals of both developing new research methods, and making practical progress in demonstrations of detection during engagements, we chose a portfolio approach where multiple techniques are pursued as possible solutions to the same problem. This allowed us to pursue risky new research with potentially high rewards simultaneously with more established methods that provide a higher level of certainty but perhaps less upside when successful.

In section 3.1, we describe our characterization of the research problems and the various research methods pursued under each topic—whether or not the methods were successful and integrated into the ADAPT system. Section 3.2 describes the methods and procedures as integrated into the final system. Since there are a significant number of methods discussed, we group their evaluations together with the presentations of the methods in Section 3, and limit Section 4 to a discussion of the results of the system as a whole.

3.1 Research Problem Characterization

Given the problem space of APT detection in the context of a high-volume of data, we have been pursuing multiple methods to address the sub-problems of: Feature Extraction, Signal Detection, and Combination & Prioritization. In each category, our team pursued strategies that encode expert knowledge and/or automatically learn from the data.

3.1.1 Feature Extraction

The Feature Extraction problem is one of making the right observations on the data to feed into the next step. As those latter steps evolved, so did the methods for producing the appropriate observations. However, all of our approaches begin by building the incoming data into a single unified graph, and storing it in a graph database.

Data was structured into a graph with each CDM record type constituting a node in the graph. Fields in a CDM record which semantically represented the identifier of another CDM record were structured as edges, with the edge label corresponding to the field name. Consequently, the nodes represented in the graph were all the Subjects, Events, and Objects (of all varieties) as defined in the CDM format, and the edges connecting them were derived from fields on Event nodes such *predicateObject*, *predicateObject2*, *subject*, and other fields pointing to other UUIDs.

With the graph of all system activity built and being updated with new data streaming in, the process of extracting features for downstream components could be run. Depending on the downstream component, the features were represented either as numerical vectors, or as (*subject*, *predicate*, *object*) tuples, where *subject* is a process node in the graph, *predicate* is an Event node connected to the *subject*, and *object* is a file, network, or other object connected to the *predicate*.

Feature vectors for the more traditional anomaly detection methods described below are produced by defining queries on the graph which traverse data according to an expert-defined policy. The queries yield a set of nodes, which are reduced variously to produce

counts or other numeric measures, encoded positionally in the vector. Vectors produced in this manner are aggregated into a matrix and the matrix used for batch processing anomaly detection techniques.

Semantic tuples of (*subject*, *predicate*, *object*) are produced for the newly developed categorical anomaly detection methods described below. Depending on the policy defined by the expert, these tuples are either used in their *triple* form, or else combined together into more complex quintuples of (*subject*, *predicate*, *object*, *predicate, subject*) when two triples share the same *object*, or (*object*, *predicate*, *subject*, *predicate*, *object*) when two triples share the same *subject*. There was also one special case of (*subject*, *subject*) tuple collection to examine anomalies related to the structure of the process tree.

3.1.2 Signal Detection

Given data structures as described in the preceding section, the second major problem is to detect meaningful signals in the volume of data under investigation. We explored many different methods for this problem from the extant literature as well as creating several new methods as described below.

3.1.2.1. Sequential Feature Explanation

Anomaly detectors typically produce a ranked list of statistical anomalies, which are then examined by human analysts in order to extract the actual anomalies of interest. Unfortunately, most anomaly detectors provide no explanations about why an instance was considered anomalous. To address this issue, we developed a feature-based explanation approach called sequential feature explanation (SFE) to help the analyst in their investigation [Siddiqui et al., 2015; Siddiqui et al., 2019]. We demonstrated on benchmark and real-world data that the explanations were often quite simple and provided insight.

3.1.2.1.1. Method Description

In order to reduce the analyst's effort for detecting anomalies, we propose to provide the analyst with sequential feature explanations (SFEs) that attempt to efficiently explain why a point was considered to be an outlier. A length k SFE for a point is an ordered list of feature indices $E=(e_1,\ldots,e_k)$, where $e_i \in \{1,...,n\}$. The intention is that features that appear earlier in the order are considered to be more important to the high outlier score of a point (e.g. $x_{(e_1)}$ is the most important). We will use the notation E_i to denote the set of the first i feature indices of E. Also, for any set of feature indices S and a data point x, we let x_S denote the projection of x onto the subspace specified by S.

Given an SFE E for a point x, the point is incrementally presented to the analyst by first presenting only feature $x_{(E_1)}$. If the analyst is able to make a judgement that x is anomalous based on only that information then we are finished with the point. Otherwise, the next feature is added to the information given to the analyst, that is, the analyst now sees $x_{(E_2)}$. The process of incrementally adding features to the set of presented information continues until the analyst is able to make a decision. The process may also

terminate early because of time constraints; however, we don't study that case in this work.

For normal points, the incremental presentation of SFEs may not help the analyst more efficiently exonerate the points. In contrast, for anomalies, it is reasonable to expect that an analyst would be able to detect the anomalies more easily by considering a much smaller amount of information than they would have to without the SFE, which should reduce the chance of missed detections. To clarify further, we assume the analyst has the expertise to decide with certainty whether an instance is an anomaly from the entire set of features if given enough time. However, the effort required to determine the anomaly may be large if all the information is shown at the start. Further, it is assumed that if the minimal set of features responsible for the anomaly are shown, the effort may be reduced (they see the minimal number of feature interactions). Hence, we assume that the amount of analyst effort is a monotonically increasing function of the number of features considered. This motivates measuring the quality of an SFE for a target by the number of features that must be revealed to an analyst for correct detection. More formally, given an anomaly point x, an analyst a and an SFE E for x, the minimum feature prefix, denoted "MFP"(x,a,E), is the minimum number of features that must be revealed to a, in the order specified by E, for a to detect x as an anomaly. The analyst may very well consult other information during an investigation. The hope is that simple and good explanations will allow the analyst to efficiently direct their attention to the key external information.

While MFP provides a quantitative measure of SFE quality, its definition requires access to an analyst. This complicates the comparison of SFE computation methods in terms of MFP. Our work, developed techniques that use surrogate models of an analyst in order to compute SFEs that aim to minimize the MFP. Complete details are in the full paper [Siddiqui et al., 2019]. In summary, developed a provably optimal approach based on the branch-and-bound algorithmic paradigm. However, the worst-case running time of that algorithm is exponential, though in practice it is often much faster. We also showed that, in the worst case, it is unlikely that there is a more efficient optimal algorithm, since the MFP optimization problem is NP-hard. In response to the hardness result, we also proposed three greedy algorithms: independent marginal, sequential marginal, and dropout. The first two, greedily add features to an SFE based on marginal distributions of the anomaly detector and the last greedily removes features.

3.1.2.1.2. Method Evaluation and Discussion

We run experiments on widely used anomaly detection benchmark datasets, where the ground truth anomalies are known. We consider modeling an analyst as a conditional distribution of the normal class given a subset of features from a data point. More formally we model the analyst as a function A(x,S)=P("normal " ⊣ | x_S), which returns the probability that point x is normal considering only the features specified by the set S. We obtain this function for our experiments by using supervised learning on the ground truth dataset to learn a probabilistic classifier for the normal class. The class probability of this classifier is then interpreted as the confidence of the simulated analyst. Given this

simulated analyst it is possible to empirically evaluate the MFP of an SFE for any data point.

We evaluated seven methods for computing SFEs. These included SeqMarg, IndMarg, SeqDO (Sequential DropOut), IndDO (Independent Dropout) and BaB.100 (branch-and-bound restricted to 100 nodes of exploration). In addition, we evaluated a random explanation method. In the random case, we report the average performance across 100 randomly generated SFEs. Finally, in order to provide a lower bound on attainable performance (lower MFP is better) we consider an optimal-oracle method, OptOracle. This method is allowed access to the simulated analyst and for each number of features i computes the optimal feature subset of size i. Clearly, OptOracle represents an optimistic bound on the performance of any SFE method that is evaluated with respect to the simulated analyst.

Figure 1 shows results for these methods on 7 benchmark data sets. We can see from these results that the explanation methods tend to produce results that are closer to the performance of the oracle than random (lower is better). These results along with significant additional results in the full paper resulted in the following key observations and recommendations.

- All of the introduced SFE methods significantly outperformed randomly generated SFEs.
- The marginal methods were generally no worse and sometimes significantly better than the dropout methods.
- When using the real anomaly detector (Ensemble of Gaussian Mixture Models), we observed little to no difference between the performance of sequential versus independent methods.
- When using the oracle anomaly detector, SeqMarg significantly outperformed IndMarg which suggests that, in general, sequential methods can outperform independent methods.
- While the BaB methods were more effective at optimizing our surrogate objective in many cases, this did not translate to outperforming the best greedy method with respect to MFP.

Overall, based on our results, SeqMarg is the recommended method for computing SFEs among the methods we studied.

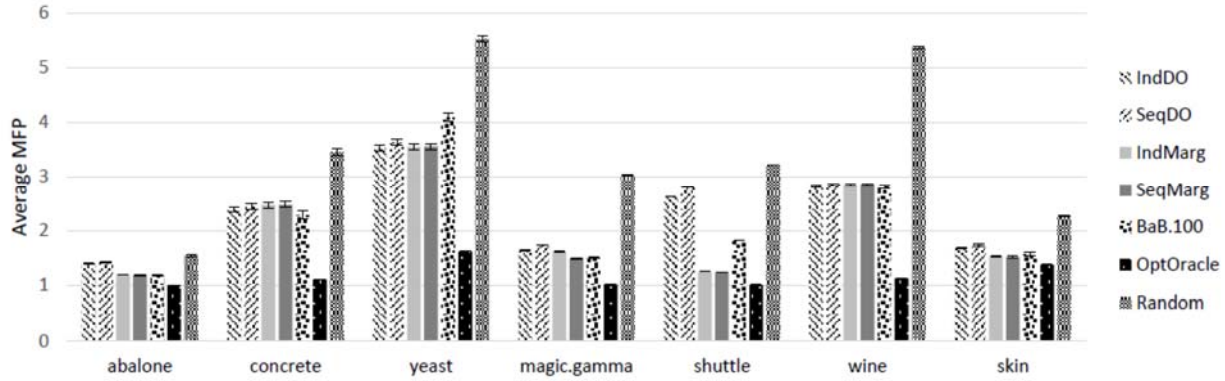


Figure 1. Performance of Explanation Methods on Benchmarks
Each group of bars shows the performance of the six methods on a standard benchmark dataset. The bars show the expected MFP averaged across anomalies in benchmarks. 95% confidence intervals are also shown.

3.1.2.2. Incorporating Feedback into Anomaly Detection

One of the issues limiting the utility of AD in practice is the large false positive rate. This is due to a mismatch between statistical and semantic anomalies. We addressed this issue by incorporating human feedback, that is, we develop a human-in-the-loop anomaly detection system which can improve its detection rate with a simple form of true/false positive feedback from the analyst [Das et al., 2016; Das et al., 2017; Siddiqui et al., 2018a; Siddiqui et al., 2018b]. We showed empirically the efficacy of the approach on benchmark problems as well as significant cyber security applications including red team attack data from the TC program.

3.1.2.2.1. Method Description

The main contribution of our work on feedback-guided anomaly detection is to formulate the problem in the framework of online convex optimization and derive simple, efficient, and effective algorithms. This is done by associating a convex loss function to each feedback response, which rewards the anomaly score for aligning with the feedback. We consider two different methods, which differ in their choice of loss function.

By leveraging the well-studied area of online convex optimization, our approaches inherit the simplicity and efficiency of the online algorithms as well as their solid theoretical grounding, where convergence is understood. Prior state-of-the-art approaches, such as AAD, are significantly more complex to implement, incur much higher computational complexity, and have less clear theoretical underpinnings. In addition, prior approaches incorporate a significant number of parameters, whereas, our approaches have a single learning rate parameter, for which we show that a single value works well across all of our experiments.

Our feedback approach can be applied quite generally, in particular, to any anomaly detector that can be represented as a "generalized linear anomaly detector", which is formally defined in the full paper. Most existing state-of-the-art detectors can be formulated within that class. Our work focuses on a particular class of generalized linear models, tree-based anomaly detectors, which includes the state-of-the-art Isolation Forest detector [Liu et al., 2018], among others. At a high-level, detectors in this class are distinguished based on how they "assign weights" to edges in their trees. Our feedback approach is able to automatically span this space, covering existing and new tree-based detectors, by tuning the weights in response to feedback.

The details of the online convex optimization (OCO) framework, which is the basis of our work, are beyond the scope of this report. Conceptually, however, the OCO framework allows us to view the feedback-guided anomaly detection problem as a game being played between our anomaly detection system and the analyst. In this game, our system makes a "move" by flagging a particular instance as an anomaly. In response, the analyst makes a "move" by indicating whether the instance is an anomaly or not. The goal of this sequential game is for our algorithm to achieve a small "regret" in performance compared to an optimal decision rule in hindsight. The power of this framework is that it makes no assumptions about the analyst and yet offers theoretical bounds on the regret. We developed several variants of the approach based on defining different types of convex "loss function" that are derived from the analyst feedback. For all of these loss functions, we derived an efficient online algorithm called Online Mirror Descent, which allows for certain constraints to be incorporated into the parameters of the tree-based anomaly detector (e.g. requiring that the weights remain non-negative).

3.1.2.2.2. Method Evaluation

We conduct experiments on individual benchmark problems from prior work and on six large anomaly detection benchmark sets. The results, in Figure 2, show that our online approaches (Linear and Log-Likelihood) are able to significantly accelerate the rate that an analyst can find anomalies compared to not using feedback. We also show significant improvements over the prior state-of-the-art method AAD.

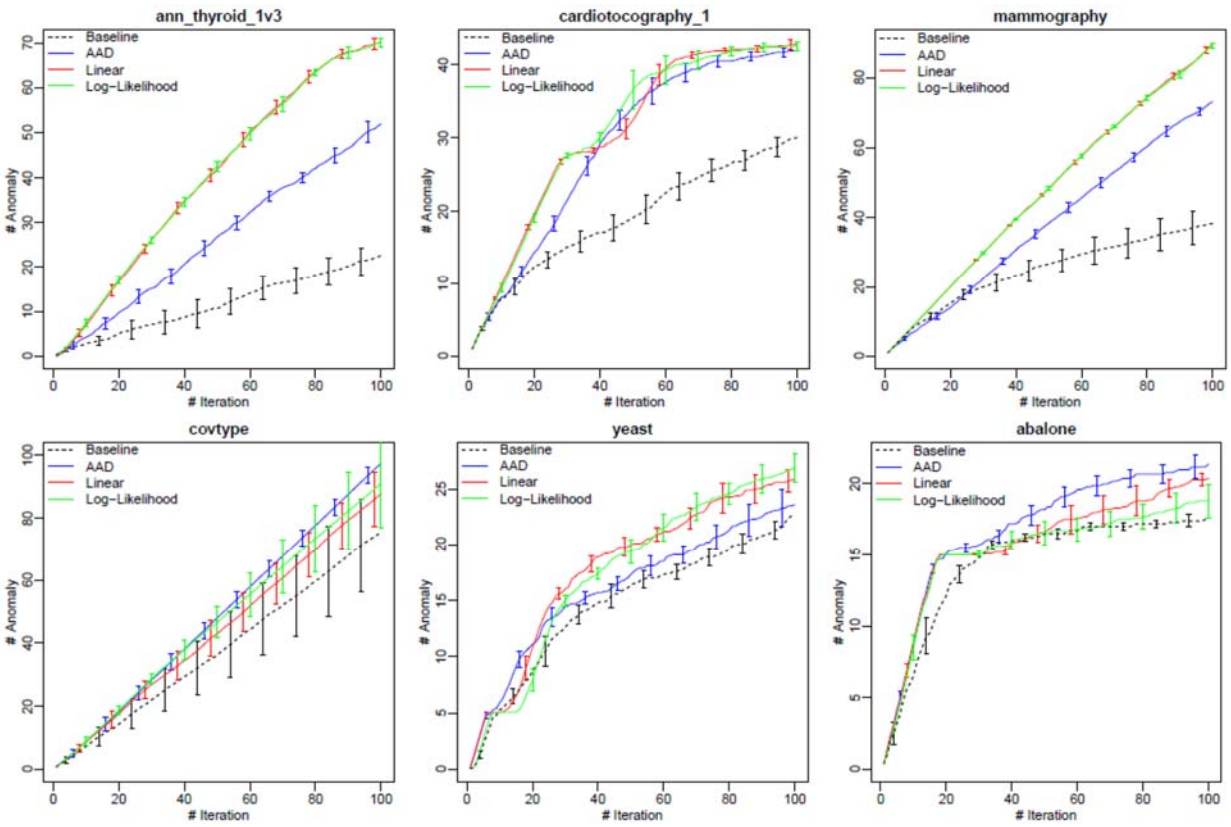


Figure 1. Feedback Result Comparison with AAD on Some Standard Anomaly Detection Datasets

Each graph shows the number of true anomalies discovered after each number of feedback iterations. 95% confidence interval are shown on each graph. Linear and Log-Likelihood are our newly proposed approaches, while AAD is the prior state-of-the-art. Baseline is the isolation forest anomaly detector that does not use feedback. This detector is equivalent to the performance of Linear or Log-Linear if they do not receive feedback.

We also demonstrate our approach on real cybersecurity attack data collected from the Engagement 3 from the DARPA Transparent Computing program. This engagement contained multiple attacks on multiple operating systems, over multiple days. We focus on the problem of detecting anomalous system entities, with the goal of quickly identifying the malicious entities that are part of an attack. The malicious entities are extremely rare in the data, which yields a very difficult anomaly detection problem. Our results in Figure 3 show that our feedback-based approaches allow for malicious system entities to be discovered significantly earlier than when no feedback is used by the system. In particular, the number of anomalies found for a given amount of feedback is generally significantly larger than for the baseline that does not use feedback.

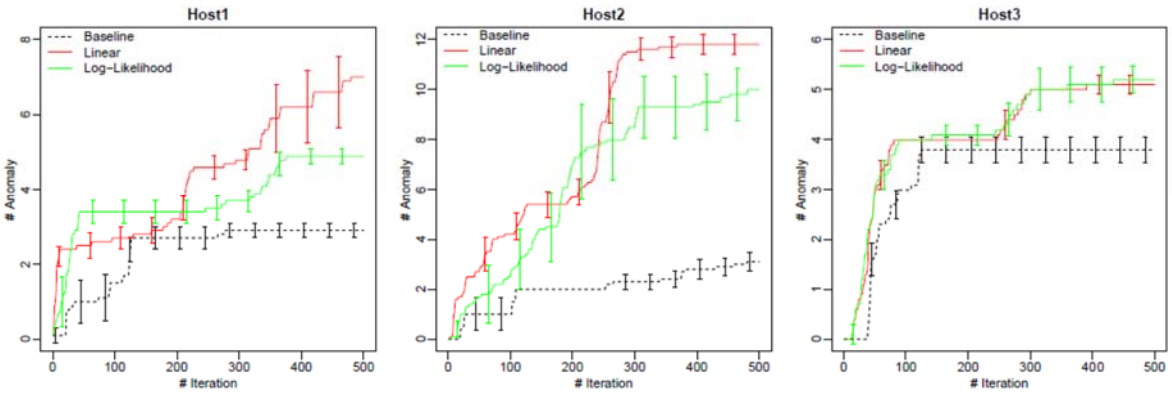


Figure 3. Feedback Iteration Results When Applied to Data From Different Host Systems

As another measure of performance on the Engagement 3 data, we measured how long it took each method to discover the first anomalous instance. We see from Table 1 that our feedback methods (Linear and Log-Likelihood) significantly reduce the time required to identify the first anomaly.

Dataset	# of iterations to 1^{st} malicious entity		
	Baseline	Linear	Log-Likelihood
Host1	42.7 ± 24.3	$\mathbf{2.7 \pm 0.7}$	12.9 ± 6.3
Host2	61.8 ± 26.9	$\mathbf{8.1 \pm 1.1}$	32.2 ± 22.7
Host3	41.4 ± 1.1	$\mathbf{21 \pm 3.5}$	$\mathbf{20.6 \pm 3.5}$

Table 1. Number of Feedback Rounds Required to Identify the First True Anomaly

3.1.2.3. Theory of Rare Pattern Anomaly Detection

From a theoretical perspective the empirical success of state-of-the-art anomaly detection algorithms is somewhat of a mystery. In particular, for high-dimensional data, all data points are anomalous in some way, suggesting that anomaly detection should not produce results better than random. Yet, state-of-the-art detectors achieve results that are far superior to random. This led us to search for a theoretical explanation for this observed performance. The result was a new theoretical framework, which captures most state-of-the-art anomaly detectors. In particular, we introduced the framework of rare pattern anomaly detection, where different detectors are characterized by their underlying "pattern space" [Siddiqui et al., 2016]. Our main theoretical results characterized the finite sample complexity of detectors in terms of properties of the pattern space. Most state-of-the-art detectors were shown to have pattern spaces that yielded polynomial time

sample complexity according to our theory. In addition, controlled empirical investigations showed that the theory predicted observed performance. This work provided the first finite sample analysis for anomaly detection in the literature.

3.1.2.3.1. Description

For moderately high-dimensional data, all data points become far apart from each other, so they are all statistical outliers in a sense. This suggests that anomaly detection by identifying outliers or distant points should perform poorly and degrade to random selection as the dimensionality grows. Empirical results, however, have shown that state-of-the-art anomaly detectors often perform quite well even for high-dimensional data. Further, these detectors tend to reach their peak performance with a relatively small amount of training data compared to what might be expected based on the dimensionality. The primary goal of this work under the Transparent Computing program was to move toward an understanding of these empirical observations by analyzing the sample complexity of a certain class of anomaly detectors.

The sample complexity of supervised learning has been widely studied and is quite well understood via the framework of Probably Approximately Correct (PAC) learning. However, this is not the case for anomaly detection, where virtually all published work has focused on algorithms with good empirical performance (with additional attention to computational speed, especially on big data sets). A key step in the development of PAC learning theory was to formalize the notion of a hypothesis space and to quantify the relationship between the complexity of this space and the amount of training data required to identify a good hypothesis in the space. We chose to follow a similar approach. Our framework is motivated by the observation that many state-of-the-art anomaly detectors can be viewed as monitoring the probabilities of certain "patterns" in the data, where a "pattern" is a subset (typically closed and compact) of the feature space. Outliers are then identified based on measures of those probabilities, where points are ranked as more anomalous if they satisfy lower-probability patterns. For example, the highly-competitive anomaly detection algorithm, Isolation Forest [Liu et al., 2008], finds outliers by monitoring probabilities in the pattern space of axis-aligned hyper-rectangles. In our analysis, a "pattern" will play the same role as a "hypothesis" in PAC learning, and the pattern space complexity will determine the number of training examples required for high accuracy.

A second key step in the development of "PAC theory was to relax the goal of finding the best possible hypothesis. Similarly, we will introduce an error parameter ϵ that determines how accurately the algorithm must estimate the probabilities of the patterns in the pattern space. Our main theoretical results show that the required sample size scales polynomially in $1/\epsilon$ (as well as in several other parameters).

We call our formulation Rare Pattern Anomaly Detection (RPAD), and an algorithm that provides PAC guarantees will be referred to as a PAC-RPAD algorithm. We prove sample complexity results for any algorithm within the RPAD framework. The framework captures the qualitative essence of many anomaly detection algorithms.

3.1.2.3.2. Discussion

We refer the reader to the full paper for technical definitions, details, and proofs. Here we give an informal presentation of the main results.

Given a space of potential inputs X a *pattern* is a subset of X, where we think of the pattern as being satisfied for elements in the subset. A *pattern space* is a class or set of patterns. Depending on whether X is finite or infinite (countable or uncountable), patterns and pattern spaces can be either finite or infinite. The complexity of a pattern space can be characterized in a number of ways, two of which are most relevant to our theory. First, for finite pattern spaces, the cardinality of the space is the key complexity parameter, noting that this cardinality is generally exponential in the dimension/size of elements in X. For continuous spaces, the analog is VC-dimension, which characterizes the ability of the patterns to "cut up" the space. For the spaces underlying many continuous-data anomaly detectors, the VC-dimension is polynomial in the dimensionality of the data.

Our basic results show that a very simple algorithm, called *RarePatternDetect*, has a sample complexity that is polynomial in the VC-dimension for continuous spaces. For finite spaces, this algorithm achieves a sample complexity that is polynomial in $\log C$ where C is the cardinality of the pattern space. The algorithm operates by first drawing a training set D of a size specified by our theory. The training set is used to estimate the normalized pattern probabilities of patterns in the space. Given a new data element x the algorithm flags it as anomalous if and only if it satisfies a pattern with estimated frequency less than or equal $\tau + \frac{\epsilon}{2}$, where τ is the specified detection threshold and ϵ is the error tolerance parameter. This test is done with an assumed subroutine $HasRarePattern$, which returns true if there exists such a pattern. For the purposes of sample complexity analysis, we will assume an oracle for *HasRarePattern*. For sufficiently complex pattern spaces, the problem addressed by *HasRarePattern* will be computational hard. Thus, in practice, a heuristic approximation will be needed, for example, based on techniques developed in the rare pattern mining literature.

3.1.2.4. Anomaly Detection for High-Throughput Discrete Data Streams

Most state-of-the-art anomaly detectors either require data to be represented via numeric vectors or are batch algorithms with poor computational complexity. There are important applications, however, where the data is fundamentally discrete and highly efficient streaming algorithms are required. This is the case, for example, in transparent computing, where the raw event data corresponds to discrete operating system events that arrive in a high-rate stream. This led us to develop a new approach to AD for streaming discrete event data. The key idea was to draw on the classic approach of Prediction by Partial Matching (PPM) for data compression, adapting it for the purposes of AD. To the best of our knowledge this has not been considered previously in the literature. This approach has not yet been published, but has been fully implemented in the ADAPT system and used for both TC engagements 4 and 5. The high-scalability and very promising empirical results in the engagements give us confidence that the PPM approach is worth significant further attention. We are currently writing a paper to be submitted to an upcoming conference.

3.1.2.4.1. Method Description

There is a body of research that uses anomaly detection for discrete sequences as intrusion detection systems [Chandola et al., 2010]. Many of these systems are insensitive to certain structural elements of the sequences such as the type of the symbols comprising the sequence or the potential presence of signature attack patterns. Further, they are often prohibitively computationally expensive in the domain of online threat detection using high-rate streams of event data. Our PPM tree anomaly detection method efficiently builds and maintains a model of streaming event data that exploits the data's structure by treating events as sequences of typed symbols and exploits the existence of common attack signatures by formulating trees specifically for detecting these attack signatures.

In our method, events are turned into specifically-ordered sequences of typed symbols by functions we call descriptors. The symbols are often the names or UUIDs of processes or files found in the input events. The symbol order and sequence length of a descriptor's output determines the structure of a corresponding PPM tree which is used to keep track of the frequency of each symbol's appearance and the conditional probabilities of a symbol's appearance given the preceding symbols. PPM trees are useful in the anomaly detection context because they have a special escape node that indicates when their descriptor produces a never-before-seen sequence of symbols. We call these sequences novelties. The escape node further helps us assign a novelty score to each novelty. A novelty can be turned into an alarm by thresholding on the novelty's score. We use a quantile threshold which allows for a flexibility in which novelties are considered anomalous that is responsive to changes in the distribution of novelty scores as the event data stream is read. Among other benefits, quantile thresholding allows us to control the rate at which novelties are flagged as anomalous. Results show that these novelty alarms are sufficient for capturing the attacks perpetrated during the transparent computing engagements.

While sufficient for detecting attacks, even with quantile thresholding, the set of novelty alarms produced during the engagements can contain a large number of false positives. This motivates an approach that can harness the sensitivity of novelties to attack-related activity while reducing the false positive rate. We examined the effectiveness of techniques that aggregate novelties according to the type of system entities involved over a specified interval of time. For instance, during stream ingestion, one version of these aggregation techniques looks at all the novelties produced during the most recent fifteen-minute interval, groups the novelties by the name of the process involved (if there was a process involved), and ranks the processes active in that interval by the number of novelties with scores below the threshold. This interval-scoped list is combined with an historical list of processes ranked in the same way. Another aggregation quantile threshold is used to raise alarms on processes that rank highly on this historical list. This and other aggregation techniques help us achieve three goals. They focus the anomaly detection on an entity type (i.e. processes) whose behavior constitutes the vast majority of attack-related behavior. They help restrict the alarms to those entities with only the most novel behavior (i.e. those processes with the most low-scoring novelties). Finally, they reduce the false positive rate compared to our non-aggregation techniques while maintaining good true positive rates.

3.1.2.4.2. Method Evaluation

We have preliminary results pointing to how well the PPM tree anomaly detection method might be expected to perform on real-word data. Four separate analyses were conducted using six (three pairs of two) of the datasets from engagement four. These six datasets were chosen because out of all the datasets for engagement four, there is a reasonably accurate basis for establishing their ground truth using a keyword matching technique based on keywords pulled from TA5 reports. The first analysis looks at the effectiveness of novelty alarms by varying the novelty quantile threshold in order to produce a Receiver Operating Characteristic (ROC) curve plotting the true positive rate (y-axis; number of true positive alarms divided by the number of attack-related novelties) against the false positive rate (x-axis; number of false positive alarms divided by the number of benign novelties). See figure 4 below. The results indicate that for at least two of the datasets (Cadets-A and FiveDirections-B) there are novelty threshold values that produce good anomaly detectors (represented by the sharp bends near the origin of the x-axis). The quality of the detectors for the other datasets, however, is too variable to consider novelty-alarm-based detection to be very reliable.

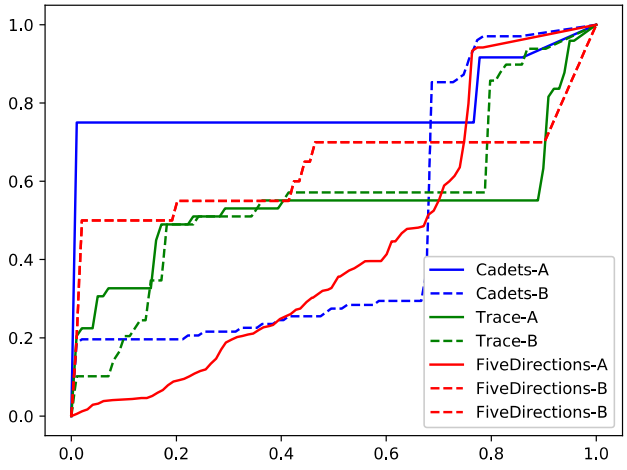


Figure 4. ROC Curves Generated Using Novelty-Based Alarms

The next three analyses look to improve upon the results of the first by plotting ROC curves based on aggregation alarms. Figure 5 shows the results of these analyses side by side. The leftmost plot shows the detectors produced by aggregating over process UUID and varying the aggregation quantile threshold. This version of aggregation produces detectors that may be considered reasonable if the goal is to keep the false positive rate down at the expense of capturing all the attack-related processes. More generally, we can see how this technique is more consistent across datasets which speaks well of the performance of aggregation techniques over purely novelty-based ones. The middle plot shows the results of the same analysis but for aggregating over process name rather than UUID. Detector performance for the Trace and FiveDirections datasets are improved but the performance for the Cadets dataset is worsened. Finally, the rightmost plot shows the

best curves possible for each pair of datasets. These curves are chosen from an assortment generated by varying the aggregation type and parameters to the anomaly detection methods not discussed here due to space constraints. The best detector appears on the Cadets-A curve where there is a point with a near-zero false alarm rate while accurately detecting almost 70% of the attack-related processes. The next best for Cadets-B has a similarly low false positive rate with an almost 60% true positive rate. The best detectors for the Trace and FiveDirections datasets are able to achieve similar true positive rates but with higher false positive rates.

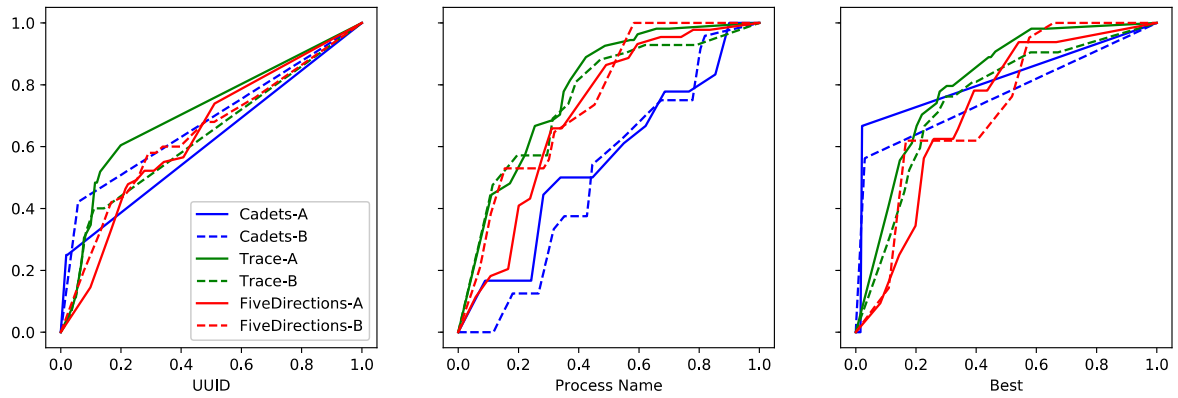


Figure 5. ROC Curves Generated Using Aggregation Techniques

Overall, we can see that the PPM tree anomaly detection method and the concept of novelty we have defined in this context can be useful as part of an intrusion detection system. The aggregation techniques take this usefulness one step further by pointing towards the possibility of developing detectors with reasonable true positive rates and very low false positive rates.

3.1.2.5. Contextual Anomaly Detection

In this section we summarize methods used and results obtained following our strategy of unsupervised anomaly detection based on Boolean contexts. (From now on, we will just use the term "contextual anomaly detection" to refer to this approach.) There are a number of algorithms already available for this problem, and we have experimented with new ways of combining existing anomaly detectors and new algorithms based on rule mining.

By "context" we just mean a dataset relating objects (typically via object identifiers such as UUIDs) and true-false attributes. Tabe CNTXT gives an example of a transactional database that is represented as a context. Such a dataset can be represented in a number of different ways: as a bipartite graph, as a binary relation R relating objects and attributes, as a Boolean matrix, as an adjacency list, as a CSV file, etc. Typically, and exclusively in this report, the objects of a context are (UUIDs of) operating system process, that is, ADMSubject nodes, and the attributes are some true-false properties (usually definable via a simple graph query), such as "did the process ever perform an event of type T?",

"was the process's executable name 'nginx'?", or "did the process ever connect to IP address 128.2.1.2?".

Objects / Attributes	*a*	*b*	*c*	*d*	*e*
UUID1	o	1	1	0	1
UUID2	1	0	1	1	0
UUID3	1	1	1	1	0
UUID4	1	0	0	1	0
UUID5	1	1	1	1	0
UUID6	1	0	1	1	0

Table 2: An Example of a Formal Context Representing a Group of Objects and Their Attributes

In this standard formal context representation, we consider an item as a property (feature) of an object and an itemset as a set of items. The image of an itemset corresponds to the set of objects including the itemset. For instance, from our previous toy example in Table 2 the image of the itemset {a,b,c} is the set of objects {UUID3,UUID5} that has a length of 2. A support of an itemset is interpreted as the relative number of objects included in its image divided by the total number of objects in the context. The support of the previous itemset {a,b,c} is then 2/6 (or 33% as relative frequency). By setting up a support threshold, one can either extract frequent patterns, which are itemsets whose support is greater than this threshold, or rare patterns whose support is below this threshold. For instance with a support threshold that equals 20%, we can observe the rare itemset {b,c,e} since its support is ⅙=16% that is smaller than 20%.

We considered in our work a number of contexts focusing on processes, and experimented also with contexts focusing on files, netflows, and other CDM objects. However, we have a much more complete picture of what is possible with anomaly detection using process-centric contexts and focus on these results exclusively in this report. The process-centric contexts we will discuss are:

- ProcessEvent (PE): in which each process is associated with the CDM event types of all events it ever performed.

- ProcessExec (PX): in which the process is associated with its executable name(s) if any - in CDM or ADM there can be more than one or zero.

- ProcessParent (PP): similar to ProcessExec but associating a process with its parent's executable name(s) if any.

- ProcessNetflow (PN): in which the process is associated with the (remote) IP addresses and port numbers of all network flows it interacts with.

- Finally we sometimes also considered a "combined" context, ProcessAll (PA) which just collects all of the above attributes into a single context, renaming to avoid confusion between e.g. ProcessExec and ProcessParent.

Although our use of the term context derives from our initial focus on a data mining technique called Formal Context Analysis (FCA), which analyzes contexts in the form of binary relations, we prefer this term over more generic terms even in other settings because it also emphasizes that we are making choices (informed by domain knowledge) about what aspects of a process's recorded behavior are significant, and serves as a reminder that the information available in a context is an incomplete subset of the process's whole behavior.

3.1.2.5.1. Baseline performance of existing algorithms

To assess the potential of the contexts mentioned above as a basis for anomaly detection, we obtained implementations of several existing categorical anomaly detection algorithms and re-implemented a few others. Some of this work was done by Mookherjee as part of a Master's student project [Mookherjee 2018], and additional results are summarized in a technical report [Berrada et al. 2019]. In this section, we briefly review the algorithms considered and the headline results.

We considered the following five algorithms from the literature:

FPOutlier (FPOF) [He et al. 2005]: This algorithm starts by mining frequent itemsets according to a support parameter minsupp. Then each object is assigned a score corresponding roughly to the number of frequent itemsets it contains. Thus, larger scores correspond to more occurrences of frequent itemsets, meaning that anomalous objects should have low scores. This approach seems well-suited to detect anomalies corresponding to expected, but missing, activity. However, objects that have unusual activity but also display a large number of common patterns may have high scores and not be considered anomalous. In addition, the fact that this approach has a tunable parameter is problematic in an unsupervised setting, since it means that we need to guess an appropriate value for this parameter in advance. We reimplemented FPOutlier using standard itemset mining libraries.

Outlier Degree (OD) [Narita and Kitagawa 2008]: This algorithm also starts by mining frequent itemsets as well as high-confidence rules, so there are two parameters, minsupp governing the minimum support of the itemsets and minconf governing the minimum confidence of the rules. Then each object is scored by applying the high-confidence rules to it, and assigning a score corresponding roughly to the difference between the object's actual behavior and expected behavior (according to the rules). For example, if X Y is a high-confidence rule and object O displays behavior X but not Y, this rule violation will contribute to the score. High scores correspond to larger differences between actual and expected behavior, so are more anomalous. Like FPOutlier, this approach seems more

likely to consider missing, but expected, behaviors to be anomalous, and could miss anomalies that consist of rare behaviors that do not occur frequently enough to participate in rules. Also, the presence of two tunable parameters is even more problematic from the point of view of unsupervised anomaly detection. We reimplemented OD using standard itemset and rule mining libraries.

Attribute Value Frequency (AVF) [Koufakou et al. 2007] is based on frequency analysis of the individual attributes independently. For each attribute the maximum likelihood probability of the attribute value assuming value 0 or 1 is estimated (in one scan over the data). Then each object is scored by adding together the probabilities of its actual attribute values. Alternatively, this algorithm can be performed over streaming data, by maintaining the probability estimates incrementally, and scoring each new object based on the current estimates before updating them [Tan et al. 2013]. AVF, unlike the previous methods considered, does not have any parameter settings. Both batch and streaming AVF are easy to implement.

One Class Classification by Compression (OC3) [Smets and Vreeken 2011] is based on a compression technique for identifying "interesting" itemsets, implemented using the Krimp algorithm [Vreeken et al. 2011]. Essentially, the idea is to first mine frequent itemsets from the data, and then identify a subset of the itemsets that help to compress the data well. Then, each object is assigned an anomaly score corresponding to its estimated compressed size. If the compression algorithm has done a good job, then objects exhibiting commonly occurring patterns will compress well, and anomalies will not. OC3 can take a minsupp support parameter, but parameter tuning is typically not necessary because the compression algorithm will filter out any non-useful itemsets; therefore we used the smallest possible minsupp setting in our experiments. The implementation of Krimp is available and we modified it slightly to perform OC3-style anomaly scoring.

CompreX [Akoglu et al. 2012] is one of the most sophisticated approaches to categorical anomaly detection studied to date. It is based on compression, like OC3, but uses a different compression strategy. CompreX searches for a partition of the attributes such that each set of attributes in the partition has high mutual information, so that compressing the attributes using Krimp is more effective than compressing the attributes independently. Since there are exponentially many partitions to consider, CompreX starts with the finest partition (all attributes are in their own class) and greedily searches for pairs of classes to merge. CompreX has no tuning parameters and was shown experimentally to be competitive or superior in anomaly detection performance to Krimp/OC3 on several datasets. However, CompreX's default search strategy is quadratic in the number of attributes; therefore, it was not usable on contexts with over 20-30 attributes.

3.1.2.5.2. Evaluation metrics

To evaluate the effectiveness of different algorithms, we considered two metrics:

- The Receiver Operator Characteristic (ROC) curve of a ranking is obtained by plotting the number of true positives against the number of false positives

observed so far. The area under the ROC curve (AUC) is a common measure of the effectiveness of ranking techniques, as well as classifiers that produce a numerical score rather than a Boolean result value. The AUC value is between 0 and 1; 1 is best, indicating that all attacks are ranked highest above all other activity. AUC is widely used in previous work on categorical anomaly detection, but in our setting we found that it was not a reliable indicator of good attack detection performance: for example, if there are 100,000 objects and only 10 of them are attacks, then a ranking that includes them all at positions 991-1,000 will have AUC of around 0.99, whereas a ranking that includes nine attacks in the top 10 but misses one attack entirely will have AUC score of at most 0.9.

- To gain an alternative perspective, we also considered the normalized discounted cumulative gain score, or nDCG. Roughly speaking, each attack found contributes to the nDCG score, but attacks found near the top of the rank ordering count much more than attacks found far down the list. nDCG seems to be an appropriate measure to evaluate anomaly detection algorithms that produce a rank ordering, since it was originally introduced for evaluating information retrieval algorithms: the higher the nDCG score, the more "hits" are found in the first few pages of a search result. nDCG is also a score between 0 and 1, with 1 being ideal behavior, but nDCG places greater weight on finding at least some attacks near the top of the ranking and does not penalize missing a single attack as heavily as AUC.

In this report, we consider nDCG scores in evaluation only.

3.1.2.5.2.1. Batch evaluation

Of the various engagements, we only have detailed, machine-readable ground truth annotations for Engagement 2 data. We have also manually constructed partial ground truth annotations for E3 data but these have not been validated or used in all of our experiments.

The following table summarizes the nDCG scores obtained by the different batch algorithms for the ProcessEvent (PE) context (using E2/Pandex data):

Source	FPOF	OD	OC3	CompreX	AVF
5dir	0.20	0.20	0.30	**0.60**	**0.60**
CADETS	0.20	0.19	0.44	**0.54**	0.51
TRACE	0.18	0.18	**0.39**	0.30	0.27
ClearScope	0.29	0.33	0.74	0.82	**0.85**

Table 3. nDCG Scores Obtained by the Different Batch Algorithms for the ProcessEvent (PE) Context (using E2/Pandex data)

We have highlighted the best (highest) nDCG score in each row in bold. The best performance was obtained by AVF, CompreX or OC3 in each case. The FPOF and OD approaches were never competitive, and interestingly, often had similar nDCG scores even though their scoring approaches seem rather different. For these two approaches, we tried a range of support and confidence parameter values and took the highest nDCG score observed (thus, these results are best-case for these algorithms).

Similarly, for the ProcessAll context (again E2/Pandex data), the results were as follows:

Source	FPOF	OD	OC3	CompreX	AVF
5dir	DNF	DNF	**0.64**	DNF	0.53
CADETS	0.21	0.19	**0.70**	DNF	0.52
TRACE	0.18	0.18	**0.46**	DNF	0.30
ClearScope	0.31	0.34	0.68	DNF	**0.83**

Table 4. nDCG Scores Obtained by the Different Batch Algorithms for the ProcessAll (PA) Context (using E2/Pandex data)

Here DNF means that the approach did not finish within an hour. For most of the approaches, run time was a few minutes at most (running over datasets representing several days' activity). Interestingly, in this case CompreX could not finish for any context, and FPOF and OD did not finish on the Windows dataset, but again did not perform well even when they were able to run. The best performance was obtained by either OC3 or AVF; what is surprising about this is that AVF is much simpler than all the other techniques, including OC3, yet it is competitive in three out of four cases, whereas OC3 only has a definitive advantage for the TRACE data.

3.1.2.5.2.2. Streaming evaluation

As noted above, AVF is straightforward to modify to a streaming, or one-pass, algorithm. On the other hand, OC3 would require a considerable effort to run in a fully streaming setting. Because AVF exhibited competitive performance compared to OC3, it is interesting to compare the streaming and batch versions to see if streaming operation has a negative impact on anomaly detection performance. We would expect to see some differences, because in a streaming setting there is likely to be an initial "warm-up" period when all activity is new and anomalous.

We compared batch AVF with streaming AVF modified to allow for different window sizes, e.g. updating/scoring after each 1%, 5%, 10%, or 25% of the data. In each case, we considered ten different randomly-shuffled orders in which to process each dataset and report the resulting average nDCG score. The results for the ProcessAll context are as follows (again for E2/Pandex data):

	5dir	CADETS	TRACE	ClearScope
Stream 1%	0.52	0.52	0.30	0.83
Stream 5%	0.49	0.52	0.30	0.83
Stream 10%	0.52	0.52	0.30	0.83
Stream 25%	0.50	0.52	0.30	0.83
Batch	0.53	0.52	0.30	0.83

Table 5. Comparison of Batch AVF with Streaming AVF for the ProcessAll Context

As these results show, streaming in blocks of as little as 1% of the data does not negatively affect detection performance for three out of the four datasets. One dataset, FiveDirections, had some variation but the worst-case degradation in nDCG score was small (from 0.53 to 0.49). These results suggest that streaming anomaly detection can be competitive with batch techniques, but there is still much room for improvement since for example the best nDCG score obtained for TRACE by AVF is much lower than the OC3 score of 0.47.

A technical report containing a more complete set of results and additional discussion is available online [Berrada et al. 2019].

3.1.2.6. Formal Concept Analysis and Frequent-Rule Mining

While the anomaly detection techniques described above help us detect outlier objects whose behavior is likely to be malicious, these techniques do not easily allow us to single out what exact pattern of behavior really lead to objects being flagged as outliers and what pattern constitutes the malicious bit. Unless we go back to the original data, it is not easy to explain what makes each flagged outlier an anomaly.

So we needed anomaly detection methods that could more easily provide some sort of anomaly explanation. Rule mining-based anomaly detection techniques seemed like a plausible candidate for that.

As their name suggests, rule mining techniques learn rules from the data (in our case, process-centric contexts) and each entity in the data is then compared against the learned rules and flagged or not as an outlier. So not only would we be detecting anomalies based on the rules learned from the data, but we could also try and use the same rules to explain the anomalies found.

There are two types of rules (rare rules and frequent rules) and therefore two types of anomaly detection approaches one can take:

1. Learn rare rules (i.e rules satisfied by a small fraction of the data and are likely to describe anomalous behavior) and find entities that satisfy such rules as such entities exhibit anomalous behavior and are most likely to be malicious. This

approach is covered in more detail in the "Rare rule mining techniques" section below.

2. Learn frequent rules (i.e rules satisfied by a large fraction of the data and are likely to describe normal system behavior) and find entities that violate these rules. We describe this approach here.

The assumption here, of course, is that objects that are part of an attack constitute a very small fraction of the data and their behavior would be distinguishable from the rest of the data. Though this approach uses rules, the rules we compare process activity against are extracted from evolving snapshots of the data as opposed to static rules/signatures crafted beforehand by experts. It might be that an approach that combines the dynamically learned rules with previously learned signatures and historical rules could improve detection rates but that would require further experimentation.

The frequent rule mining-based anomaly detection approach can be summarized as follows:

1. First, mine the process-centric contexts for frequently co-occurring sets of attributes (i.e itemsets).

2. Derive association rules based on the mined frequent itemsets

3. Score each object in the context based on the rules they violate and the strength of the rules violated

This approach is similar in spirit to the Outlier Degree algorithm, but there are two important differences. First, OD extracts rules from all frequent itemsets, whereas our approach uses only closed itemsets; usually there are far fewer closed itemsets and resulting rules. Second, our approach to anomaly scoring is sensitive to how many rule violations there are and how surprising the rule violations are, whereas OD does not take into account the latter factor.

One way we can mine frequent itemsets is through Formal Concept Analysis (FCA). Here we only explain the basic principles of FCA. For more details on FCA, see [Wille 1992], [Ganter and Wille 1997] and [Ganter and Stumme 2003].

In FCA terminology, objects are rows of the data matrix i.e. context while attributes are the columns of the matrix.

More specifically, FCA searches for pairs of sets of objects (the extent) and a sets of attributes (the intent). These pairs, called formal concepts, which have the following characteristics:

- all objects in the set of objects have all the attributes in the Itemset/set of attributes

- there are no other objects in the formal context that have all the attributes in the set of attributes/itemset (of the formal concept)

- there are no other attributes in the set of attributes/itemset (of the formal concept) that all the objects (in the set of objects of the formal concept) have.

Note that the intents of the formal concepts extracted by FCA are closed itemsets.

Mining closed itemsets as opposed to all itemsets already reduces the number of possible rules we can extract from the data but there might still be significant information redundancy in the intents extracted by FCA. So instead of mining all possible concepts through FCA (which is also computationally very expensive), we also set a minimal support threshold (i.e the minimal fraction of context objects a concept extent should contain) so that we only keep frequently occurring concepts and therefore frequently occurring closed itemsets.

To illustrate this, here is a simple example. Suppose we have the following view of the data (context), with rows representing process identifiers and columns websites accessed by those processes:

id	abc.com	xyz.com	evil.com
P17	1	1	0
P42	1	1	0
P1337	0	0	1
P007	1	1	1

Table 6. An Example Context

This context contains three concepts:

- Extent: {P17,P42,P007} and Intent:{abc.com,xyz.com}

- Extent: {P1337,P007} and Intent:{evil.com}

- Extent:{P007} and Intent:{abc.com,xyz.com,evil.com}

The associated itemsets (i.e intents) are the attribute sets {abc.com, xyz.com}, {evil.com} and {abc.com,xyz.com,evil.com}. If we set a support threshold of 3/4, then the only frequent itemset in this context is {abc.com,xyz.com}.

We then use the closed itemsets to derive association rules of the type A → B (i.e A implies B where A and B are disjoint sets of attributes) and only keep the rules whose confidence (i.e likelihood of seeing B when A is seen) is above a certain threshold. We also prune the rules (to reduce redundancy) and remove rules of the type C → D where C ⊆ A and D ⊆ B (i.e we remove rules whose antecedents and consequents are included in other rules).

Finally, we go back to the original context and compare the attributes of each object (i.e each row in the matrix) has with the association rules learned and score the object depending on the rules it has violated weighted by the confidence of the rules violated. Objects are then ranked based on their score, with objects with the highest scores at the top.

Table 7 shows the performance (nDCG scores) of this approach (with the optimal support/confidence setting for each data source/context pair) on E2/Pandex data. No single context seems to perform best for all data providers, though ProcessNetflow (PN) is best for three datasets out of four (the exception being CADETS, on which it performs rather poorly). This supports the conclusions made in [Berrada and Cheney 2019], [Berrada et al.] and previous sections that it is hard to know a priori which context would perform best and that there is no guarantee that such context exists.

The results also support the conclusion that combining all the data in one dataset and applying the anomaly detection approach on it is not necessarily the best performing strategy (not to mention that it's computationally expensive): the only case where ProcessAll performs best is on CADETS data.

Source	*PE*	*PX*	*PP*	*PN*	*PA*
FiveDirections	0.26	0.15	0.09	**0.67**	0.26
CADETS	0.29	0.22	0.18	0.18	**0.31**
Trace	0.35	0.30	0.28	**0.38**	0.35
Clearscope	0.45	0.47		**0.58**	0.45

Table 7: nDCG Scores for Optimal Support/Confidence Setting for FCA-Based Anomaly Detection on E2/Pandex Data
(best performance per data source in bold, second best underlined)

For the FCA-based approach to work, two parameters need to be tuned: support and confidence. But, on E2/Pandex, for most datasets (except CADETS), the optimal support/confidence settings were relatively stable and small variations in support and/or confidence didn't seem to affect the performance. Table 8 shows the optimal support/confidence thresholds per dataset/context for E2/Pandex data.

Source	*Support*	*Confidence*
CADETS	50 (except PP: 60)	PE/PX: 90 PN: 42 PP: 80 PA: 97
FiveDirections	42 (except PN: 70)	60
Trace and Clearscope	42	70 or 80

Table 8: Optimal Support/Confidence Settings for FCA-Based Anomaly Detection on E2/Pandex Data

Compared to batch AVF, the performance of the nDCG scores produced by the FCA-based approach are generally lower but:

- the performance of the FCA-based approach does not seem to be affected by the size of the data the way that AVF does (AVF's performance seems to degrade with data size)

- the overall performance on Trace data seems to be more stable (the FCA-based approach outperforms AVF on 4 contexts out 5, PX being the notable exception)

On the other hand, the FCA-based approach can provide rules that help understand the reason for an anomaly having a high score, and performs much better than the OD approach, which also uses frequent rule mining. Both the scoring and rule pruning strategies could be affecting the scores so it might be worthwhile exploring different scoring and rule pruning strategies.

3.1.2.7. Rare Rule Mining

In this section, we present another anomaly detection approach that relies on the extraction of rare association rules from summaries of process activities and attributing an anomaly score to the processes based on those extracted rules.

As in the previous approaches, we have also used the formal context data representation as illustrated in Table 1, where a set of objects (processes) are related to their properties (operating system events or netflow activities) through a binary relationship. Our intuition was about summarizing the process activities with categorical or binary features such as the kind of events performed by a process, the process executable name, parent executable name and IP addresses. The hypothesis that we made is justified by the fact that some attacks could be performed through rare combinations of such attributes.

As explained above, the association rules have the form of an implication X → Y where X and Y are two disjoint itemsets [Agrawal et al. 94]. For instance, using the rare itemset $\{b,c,e\}$ from Table 2, the algorithm can derive the rare rule $b, c \rightarrow e$. It can be noticed that when the number of attributes in a context converges to n, then the number of itemsets tends to 2^n. Thus exploring the super set of all possible itemsets is not conceivable. This is why we have used a heuristic, which avoids the exploration of the total search space, by extracting only a reduced set of the rare itemsets called MRIs (or minimal rare itemsets) [Szathmary et al. 12]. The first step of our rare rule mining-based anomaly detection method starts by extracting these MRIs, then tries to derive association rules, which can be generated by finding combinations between those itemsets.

The rare rule mining-based anomaly detection approach that we have proposed emphasises processes that satisfy rare association rules and consequently have rare behavior in the provenance traces. An anomaly score is calculated for each process through the quality measures of the rare rules it satisfies. There exist many interestingness measures that can be used to highlight the quality of the rules (e.g. confidence, lift, coverage, etc …). In our work we have used the max of the lift as a main measure for processes scoring. The output lists of the anomalous entities are then ranked downwardly within those scores. An efficient detection would rank the most anomalous entities in the top of the list.

This approach has been evaluated using the previously described formal contexts we generated from different O.S. traces. For each operating system, anomalous processes have been ranked with their scores, and true positives have been identified using the ground truth files available with E2 data.

For better understanding the ranking results, we have proposed a visualisation technique called Band Diagrams, in which the generated ranked lists in each context/O.S. are represented as horizontal bands and the positions of the true attack processes (true positives) are highlighted with red vertical lines. Top ranked anomalous entities would be in the left-hand side of the bands, so obviously a chart with multiple red lines in that region would be considered as highly efficient.

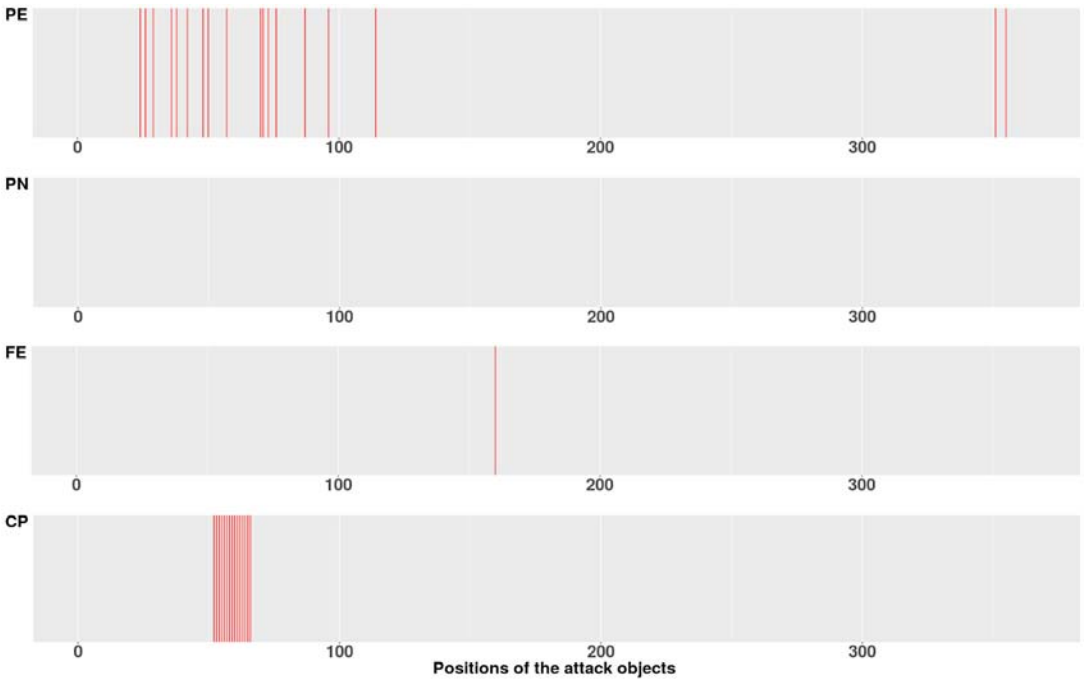


Figure 6: Example of a Band Diagram Representing Some Anomalous Processes in Different Formal Contexts
The positions (x-axis) of the attacks processes (true positives) are represented as red lines in the ranked lists.

The effectiveness of the rare rule mining anomaly detection approach has been also evaluated with the nDCG scores, and compared with several other existing approaches (FPOF, OD, CompreX, AVF and OC3).

In the following tables, nDCG scores are presented for ProcessEvent (PE) and ProcessAll (OA) contexts (using E2/Pandex data). We have highlighted the best score in each row in bold.

Source	FPOF	OD	OC3	CompreX	AVF	Rare Rules
5dir	0.20	0.20	0.30	0.60	0.60	**0.82**
CADETS	0.20	0.19	0.44	0.54	0.51	**0.64**
TRACE	0.18	0.18	**0.39**	0.30	0.27	0.13
ClearScope	0.29	0.33	0.74	0.82	**0.85**	**0.87**

Table 9. nDCG Scores for ProcessEvent (PE) Context

Source	FPOF	OD	OC3	CompreX	AVF	Rare Rules
5dir	DNF	DNF	**0.64**	DNF	0.53	0.61
CADETS	0.21	0.19	**0.70**	DNF	0.52	0.36
TRACE	0.18	0.18	0.46	DNF	0.30	**0.54**
ClearScope	0.31	0.34	0.68	DNF	**0.83**	0

Table 10. nDCG Scores for ProcessAll (PA) Context

In summary, these results have shown that the rare rule mining approach has not only been able to rank anomalous entities in the top of the output lists, but also flag the APT-like attacks in a fast running time (as reported in the next table). Moreover, the security expert could easily interpret the results by checking out the attacks through the obtained association rules, since these implications highlight rare combinations of actions that have been performed by attackers during their malicious activities.

	Winner context	Best nDCG	Running time (sec)
5dir	PE	0.82	4.60
CADETS	PE	0.64	12.18
TRACE	PN	0.58	3.53
CLEARSCOPE	PE	0.87	0.78

Table 11. Max nDCG Scores Obtained With the Rare Rule Mining Anomaly Detection Method Using Different Databases and Contexts

A paper on these results is submitted and currently under review.

3.1.3 Combination, Prioritization

3.1.3.1. Multi-context aggregations

As illustrated by the results presented so far in sections 3.1.2.6 and 3.1.2.7, the best anomaly detection performance is not always obtained with the same context. This arises naturally because some attacks involve suspicious network activity, while others involve unusual event patterns or unusual executable or parent/child relationships. This leads to a problem: if we want to make the most effective use of the results, we would like to know which context has attacks ranked closest to the top, but we do not (and maybe cannot) know this in advance, since this may depend on the attack itself.

Therefore, we investigated the problem of combining the results of several contexts into a single unified ranking. This is an instance of a more general problem called rank aggregation [Lin 2010]. There are a wide variety of possible techniques for combining rankings, some more suitable to our setting than others, and some very simple to implement while others rather involved. We experimented with several of the easy-to-implement approaches, and here report representative results for two aggregation techniques:

- Geometric mean: We construct a new score for each object by taking the geometric mean of the ranks (position after sorting by scores) of an object in each context.

- Sum of scores: We construct a new score for each object by adding together the scores obtained in each context (ignoring the ranks).

We applied these (and several other) rank aggregation approaches to E2 and E3 data, using the AVF and OC# anomaly detectors (in batch mode). Although geometric mean is meaningful for any ranking technique, the sum of scores approach only really makes sense where the scores have an additive interpretation. This is the case for OC3 (in which the scores can be interpreted as compressed sizes, so adding the scores corresponds to taking the size of concatenating the compressed representations). However, for AVF and other techniques we investigated, the sum of the scores does not have a clear interpretation, and unsurprisingly we obtained poor results when using the sum-of-scores approach with AVF. Therefore, here we just report results for E2/pandex data using OC3. These results are representative of the additional results for AVF and E3 data.

The following table shows the nDCG scores obtained using OC3 on each context (including the four base contexts and the combined context PA), and the nDCG scores results of geometric mean and sum-of-scores aggregation:

	PE	PX	PP	PN	PA	sum	geom
FiveDirections	0.30	0.28	0.21	**0.71**	0.64	**0.76**	0.40
CADETS	0.44	0.46	0.43	0.34	**0.70**	**0.76**	0.67
TRACE	0.39	0.31	0.24	**0.49**	0.46	**0.55**	0.41
ClearScope	**0.74**	0.39	0	0.66	0.68	**0.80**	0.57

Table 12. nDCG Scores Obtained Using OC3 on Each Context

These results support two interesting conclusions: First, the naive approach of combining all of the contexts into a single one (PA) often under-performs compared to the best individual context; the only case where PA's nDCG score was better than those of the four base contexts was CADETS. On the other hand, combining the scores using the sum-of-scores technique (on this dataset) always leads to better nDCG scores than those

obtained on any of the base contexts or PA. On the other hand, nDCG scores obtained using the geometric mean are not as competitive in this case. Analyzing a single large context can be considerably more expensive than analyzing several smaller ones and combining the results; our results show that the latter approach is also often more effective.

We recently published a workshop paper containing these results [Berrada and Cheney 2019], which contains more results and discussion.

3.1.3.2. Dynamic Thresholding

Many of the anomaly detection techniques described above produce a ranking of items under investigation. The ranking is determined by an anomaly score associated with each item. The decision for what to report as a final alarm becomes a question of question where in that ranked list to draw the line—or what the anomaly score threshold should be.

Leading up to Engagement 4, we developed a technique for dynamically adjusting this threshold based on the number of alarms produced so far. Given the ranked list of potential anomalies, the threshold value is adjusted up or down depending on 1.) the number of alarms produced recently, and 2.) a parameter set by the user for how "noisy" they want the system to be.

The noisiness parameter is used to determine the size of the set computed by a quantile function applied to the ranked list of alarm candidates. New alarms that rank high enough to be sorted into the top quantile are yielded downstream to be produced as final alarms (pending other methods applied later; see the next section).

For example: Given a user noisiness setting of 0.01, we might determine that only the top 1% most novel alarm candidates should lead to final alarms. Considering the current list of previously produced alarms, the lowest item that ranks in the top 1% quantile will be examined, and its anomaly score (e.g. 0.9975) is used as the new threshold for the next period of time (the time period is an adjustable setting; we used: 15 minutes). If a new alarm candidate is produced with a calculated alarm score higher than the threshold (0.9975), then the candidate is yielded downstream as a final alarm. All other alarms with scores lower than the threshold are discarded.

3.1.3.3. Population Ranking and Filtering

Before the final engagement, we developed an experimental method to attempt ranking of alarms across all detection methods. This procedure established a time span under which to collect all alarms, grouping them by the process instance implicated by each alarm. A process instance is defined as a unique tuple produced from (*processName*, *processID*), as defined by the incoming data. Process instance alarm counts were sorted in descending order and an alarm emitted only if it was in the top percentile, having had the most alarms produced for that process instance within the time window.

3.2 Implementations

3.2.1 Actor System

The Transparent Computing program was meant to address the needs of large enterprise organizations, like the Department of Defense. This goal has the automatic expectation that for a system to be usable, it must scale to cover huge numbers of systems being analyzed. For this reason, we chose to build our system on the Actor Model [Hewitt, et al., 1973], a longstanding and well-respected method for building high-performance, resilient distributed systems. Everything else in our system is built inside of the Actor System and implemented in a compatible fashion.

We chose Akka[1] as the backbone implementation library because it is written in a functional programming language (Scala) and has been used for 10 years in large scale industrial deployments. It has very high performance and low overhead, with many additional features—like stream processing—which were valuable on this program.

3.2.2 Back-pressured Ingest

Ingest of CDM data from Kafka was done using a custom Kafka ingest stream. That stream was fully back-pressured so that if the rest of the system is running slowly, the ingest of CDM data from Kafka is slowed down. Consequently, the system will never overrun the available resources—RAM or CPU—and performance of the system as a whole is correctly captured by the ingest rate.

3.2.3 Graph Rewriting / ADM

After ingesting from Kafka, the ADAPT system rewrites the CDM data into a format we called ADM—the ADAPT Data Model. This graph rewriting process was done in a fully streaming fashion. The end result was a graph of data that had the logical entities resolved into a single representation in the data, corrected for some CDM problems we had seen from TA1 providers, handled out-of-order data and missing data correctly, and raised the level of abstraction of the incoming CDM data to better match our analysis requirements.

3.2.4 Graph Data Management

Our system evolved through the use of several different graph databases, from Titan, to Neo4j, to the prototype graph database and interpreter: Quine, developed by our team outside of the Transparent Computing program. The Titan and Neo4j databases proved to be much too slow to handle the volume of Transparent Computing data being produce, even though they were the current market leaders and represented the state-of-the-art in graph data management.

[1] https://akka.io

Quine represents a fundamentally new graph database with additional novel capabilities that proved essential for the Transparent Computing program. Quine allows queries to be issued as "standing queries" which remain resident in the database and are resolved in an extremely efficient fashion using a strategy of partial evaluation. These standing queries were used to extract the behavior patterns described in the next section. Quine also does not need to create nodes before they are referenced or updated (handling out-of-order data), and handles update instructions more efficiently than the other systems. The net results allowed the Quine system to scale well beyond the requirement of the final engagement.

3.2.5 Behavioral Pattern Matching

Whether producing numeric feature vectors, or 3- and 5-feature semantic tuples, the anomaly detection is done separately for each of the following expert-defined categories (noting the tuple shape in the "Type" column):

Name	Type	Description
ProcessFileTouches	SPO	Observe all files connected to a Process node by any type of interveneing event.
FilesTouchedByProcesses	SPO	Same as ProcessFileTouches except that the conditional probabilities for the categorical anomaly detection appear in a different order.
FilesExecutedByProcesses	SPO	Observe all the path of all files connected to a process by an execute event.
FilesExecutedIshByProcesses	SPO	Same as FilesExecutedByProcesses, but also includes events which represent a more liberal notion of execution, like LoadLibrary, some MemoryMaps, and StartService events.
ProcessesWithNetworkActivity	SPO	Observe all processes with any event connecting them with network activity (IP and port).
ProcessDirectoryReadWriteTouches	SPO	Observe the processes that have a read or write event to a file object, and extract the fully qualified

		directory of the file (but drop the filename).
ProcessesChangingPrincipal	SPO	Observe processes which change the user associated with them.
SudoIsAsSudoDoes	SPO	Observe all event types and all objects connected to those events if the event is done by a process with administrator privileges.
ParentChildProcesses	SS	Observe each parent/child relationship from the process hierarchy.
FileExecuteDelete	SPOPS	Observe the processes that perform the ordered combination of an execution and deletion of a specific file. Also observe the file path.
FilesWrittenThenExecuted	SPOPS	Observe the processes that perform the ordered combination of a write and execution of a specific file. Also observe the file path.
CommunicationPathThroughObject	SPOPS	Observe the processes that perform the ordered combination of writing to an object (e.g. file, netflow, socket, pipe, etc.) and reading from that object. Also observe the name of the object serving as communication medium.
CrossHostProcessCommunication	SPOPS	Same as CommunicationPathThroughObject except that this category is specially written to navigate the (somewhat complex) structure of reciprocal network communication being done by two different processes on two different computers. This category observes the process names, and the network addresses and ports, with easy access in the exploration UI to investigate the other activity done by each process.

ProcessWritesFileSoonAfterNetflowRead	OPSPO	Observe a process which first reads from a network connection, then writes to a file withing a specifiable time period.

Table 13. Description of the Behavior Types Collected and Used as the Policies Defining Classes of Categorical Anomaly Detection During Live Engagements

Each of these categories aggregates a separate collection of feature vectors or tuples. Anomalies in these categories are found using the techniques described in Section 3.1.2, and the respective alarms combined, filtered, and sorted as described in Section 3.1.3.

3.2.6 Prediction by Partial Matching

PPM tress are created for each category described in section 3.2.5. Each tree learns the baseline of normal behavior and adjusts its threshold as described previously. For each observation passed in to the root of the PPM tree, the corresponding branches are created as nodes in the Quine database. If the branches already exist, then Quine interprets the PPM observation accordingly and adjusts the counters in the probabilistic models.

Each tree has specified conditions applied for when it filters alarms. The simplest version of these conditions is simply an existence check—if an alarm is produced because the path in the tree has never been seen before, produce an alarm. However, in the final engagement, all trees were set to use more complex criteria like the dynamic threshold filtering described in Section 3.1.3.

3.2.7 Summarization

Since an alarm by itself contains only as much data as described in a single observation (see the table in Section 3.2.5), and an analyst would need more context to understand how to process an alarm, the final component of the automated system takes the data encoded in the alarm and queries the Quine graph database to pull out the context of the accused process to better describe the behavior related to the alarm.

For some processes, the relevant data for a single alarm can be far more than an analyst can practically use. So the needs exists to summarize that large volume of data. Summarization is a considerable unsolved research topic in its own right, and beyond the scope of the ADAPT project. Our implementations on this program was limited to some fairly simple attempts to compress the data and maintain usability. Engagement 5 suffered from over-ambitious summarization, which led to correct automated alarms that were not interpretable by a human analyst. Engagement 4 had seen better summarization results.

3.2.8 Exploration and Explanation UI

The ADAPT team developed an interactive forensic analysis and explanation tool in Engagement 1 which we continued to use throughout the life of the program. This tool was a web-based UI served up by the ADAPT system for exploring the CDM/ADM

graph data. In addition to creating the visualization, development of the exploration tool also entailed defining a UI language that also continued to evolve throughout the life of the program.

The role of the UI language was to define the visual look-and-feel, but especially to encode and ever-growing library of analyst judgments about what kind of graph structures and relationship are meaningful, given the expert knowledge they have about the system. As these judgements are encoded, they become available in the UI as context menu queries available by right-clicking a node. Based on the node's type and surrounding structure, different queries are made available to the user. So the analyst has shortcuts for finding potentially complex subgraphs matching specific patterns. When the results are rendered in the UI, the complex traversal is represented with a single dotted "synthetic" edge, and the resulting subgraph potentially compressed into a single visual node, if desired. This UI language became an integral component for our team in performing research between engagements (including labeling data from Engagement 2) and performing analysis of alarms during engagements.

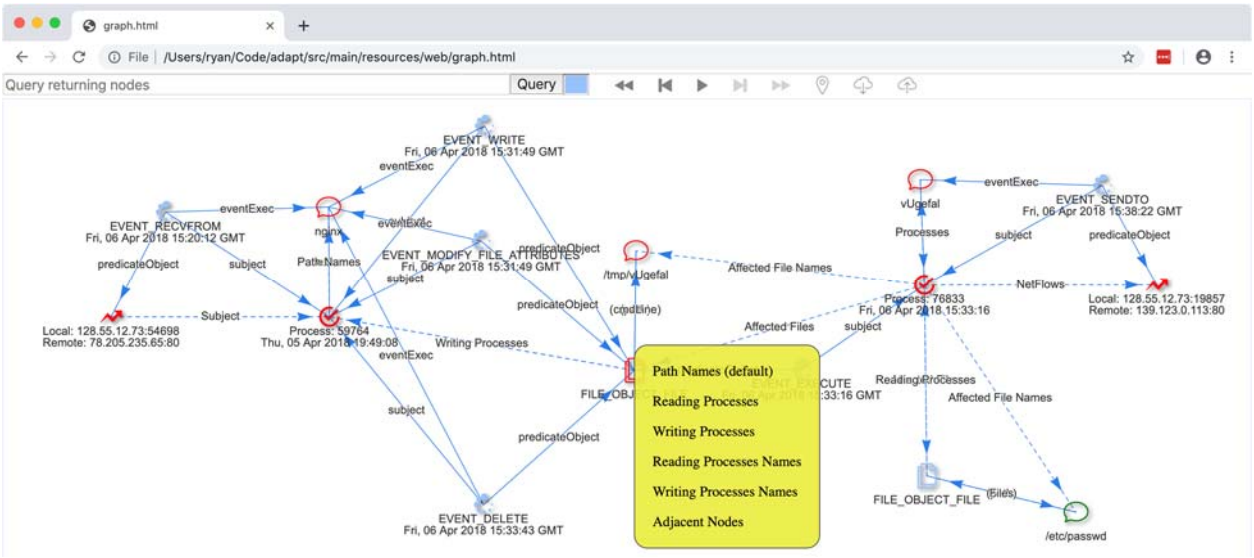


Figure 7: Example Screenshot of the Exploration and Explanation UI
It shows the dotted "synthetic" edges, and the context menu defined by the UI language making high-level semantic queries easily accessible to the analyst.

4.0 RESULTS AND DISCUSSIONS

4.1 Evaluation

Up to Engagement 3, all TA2 teams had been focusing on human-centered methods for detecting APT activity. Starting with Engagement 3, the ADAPT team succeeded in developing *fully automated* methods for successfully detecting APT activity. For the first time in the program, this allowed for a TA2 system to be evaluated objectively, where the results were produced by the system alone, and not dependent on applying human judgment or having humans revise the results delivered by the system. The possibility of objective evaluation marked a significant accomplishment in the life of the program.

Correspondingly, following Engagement 3 the ADAPT team engaged with staff at Kudu Dynamics—the TA5 evaluation team—to update the evaluation methods so that the automatic detection capabilities can be assessed. For the five months leading up to Engagement 4, the ADAPT team communicated with Kudu Dynamics and worked to set up a system to receive all of the automated alarms, where they would be persisted unchanged and available to Kudu Dynamics for inspection and evaluation.

A Splunk[2] system was used as the system-of-record to collect the automated ADAPT alarms. Kudu Dynamics confirmed access to the Splunk system as well as an objective procedure for testing whether their attack activity was correctly identified in alarms produced from the ADAPT system. The evaluation procedure involved Kudu Dynamics staff members querying the Splunk system either in real-time or any point after their attacks were performed (according to their availability). Queries in Splunk were to use the knowledge of ground truth attack activity known only to Kudu Dynamics to test whether the ADAPT system successfully produced an alarm for that activity.

Despite the mutual plans and efforts by the ADAPT and TA5 teams, after delivering their report and presentation at the PI meeting following Engagement 4, Kudu Dynamics confirmed that they did not apply the evaluation method that had been agreed upon. They were apologetic, but did not give a reason why, and asked that we deliver raw data that was published in to Splunk. We delivered that data days after their request, but noted that the incorrect evaluation results were never revised as discussed.

The evaluation failure occurred again for Engagement 5 where the process remained as agreed, but the TA5 evaluation team did not apply the agreed upon evaluation standards or methods.

As a result of these evaluation failures, two important consequences are identified:

1. The government has incorrect information about the success of automated APT detection done by the ADAPT system. We believe the Engagements 4 & 5 reports are incorrect, as well as PI meeting presentations based on those reports.

[2] http://www.splunk.com

2. Program results for the ADAPT team reported herein rely on our team applying the objective criteria previously agreed to with Kudu Dynamics. This does mean that these results represent a self-evaluation. However, since they rely only on the application of mechanical processes of looking up the data in the TA5 documented ground truth in the Splunk system-of-record for alarms produced live during the event.

4.2 Engagement Scenarios

On five occasions over the life of the program, all teams performed in the engagements to rehearse various attack scenarios. The ultimate measure was whether the analysis teams (TA2) could successfully detect attacks performed by the TA5 teams.

The idealized TA2 performance might be hoped to show linear progress through each engagement, steadily increasing attack detection with each new round. As Engagements progresses, this proved unrealistic since the TA1 data providers were continually updating their systems, and the TA5 teams continually updating their attacks. Consequently, there was no linear progression of detection results to show—with the ADAPT system's best results being 100% detection in Engagement 3.

We did achieve consistent additive progress to ADAPT system capabilities as the Engagements progressed. The following subsections briefly describe those accomplishments.

4.2.1 Engagement 1: Initial architecture, interactive exploration UI for forensic analysis and explanation

The work leading up to Engagement 1 included cross-team efforts to define the Common Data Format (CDM) that all teams would use for data interchange. Concurrently with the data syntax design, a plan for the combined Engagement architectures was settled on, using Kafka as the persistent log, using only a single partition for each TA1 data source (which becomes the bottleneck by Engagement 5).

Upon ingesting the CDM data, our team successfully demonstrated our ability to do forensic analysis using the exploration/explanation UI described in Section 3.2.8.

4.2.2 Engagement 2: Human/Machine team analysis framework for focusing analyst attention and providing feedback into the system; data labeling.

For this engagement, we build on the prior system to incorporate anomaly detection methods to score and rank all process activity. The scored alarms were sorted and presented to a human analyst to consider whether to accuse the implicated process as an APT. The human would record their assessment of the suspicion and feed that back into the system. This served as a baseline for experimentation with using feedback to train the anomaly detection, but the overall problem of associating *anomalous* activity with *suspicious* activity remained outstanding.

Following Engagement 2, the ADAPT team spent considerable time labeling the CDM data to annotate which CDM IDs were part of the attacks performed by the TA5 team. This human labeling was very time consuming, but produced the only baseline on the program for labeled training data for supervised machine learning.

4.2.3 Engagement 3: CDM translation to ADM, combined-OS analysis, experimental automated detection.

During Engagement 3, the ADAPT system reached its maximum resource usage. The data production rates were also much higher on this engagement than previous engagements (or expectations as described by TA1 teams). The increased load surfaced some configuration problems with the virtual machines provided by the TA3 team for operation of the ADAPT system. As a result, the ADAPT system did not run in real-time during the engagement. The VM configuration problems were fixed after the engagement concluded and the ADAPT system run on the same data.

Since this engagement also was our first implementation using automated detection, there needed to be no human analyst interpreting the results. This allowed us to preserve the logic of analysis and run the same procedure at any time. Since no human is involved in the production of alarms, no bias—or foreknowledge of the attacks performed—can influence the final results. Consequently, the script created to test whether the attack components described in the TA5 ground truth appear in alarms produced by the ADAPT system represents an objective evaluation.

Machine analysis of the automatically produced alarms allowed the ADAPT team to analyze all alarms produced by the ADAPT system and classify them as either true or false alarms, providing for the first time in the program: a measure of the cost of achieving the true alarms. The more false alarms created, the higher the cost of producing the true alarms. We continued to use this scripted objective analysis for future engagements, but to our knowledge, no other teams have produced any measure of the cost to human attention which their methods require.

The results of the scripted analysis of automated alarms produced by the ADAPT system for Engagement 3 is reflected in the following table:

Overall	ADAPT		
	Total Attacks	*Total Components*	*True/False Positive*
CADETS	4/4 100.0%	58/91 63.7%	220/1605 12.1% True Pos.
FiveDirections	3/3 100.0%	42/67 62.7%	313/1157 21.3% True Pos.
TRACE	5/5 100%	61/73 83.6%	234/3267 6.7% True Pos.
THEIA	5/5 100.0%	63/79 79.7%	393/667 37.1% True Pos.
ClearScope	3/3 100.0%	32/46 70.0%	41/1196 3.3% True Pos.
Total	**20/20 100.0%**	**256/356 71.9%**	**1201/7892 13.2% True Pos.**

Table 14: Results of a Scripted Analysis of Automated Alarms Produced by the ADAPT System in Engagement 3
Classification of false positives represent the only measure of the cost to human attention achieved on the program.

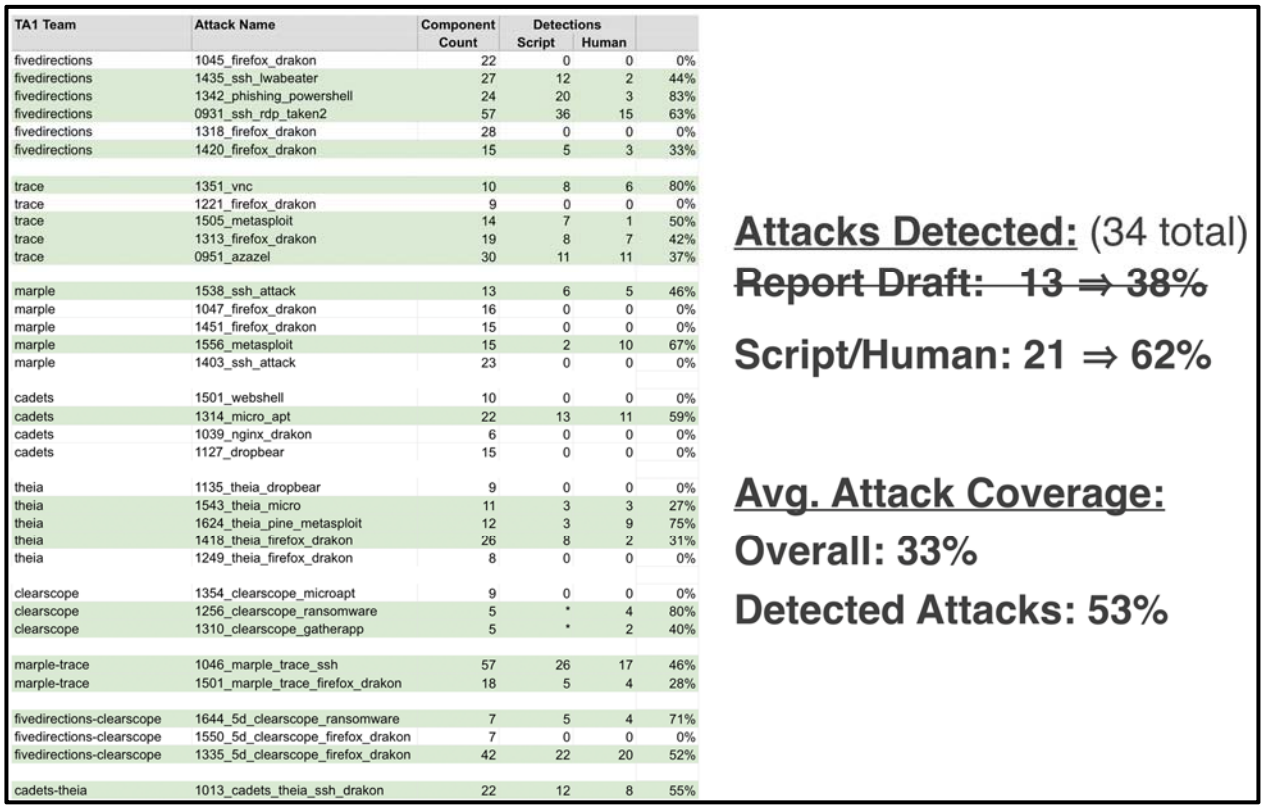
Engagement 3 was also the first time our system rewrote the incoming CDM graph into ADM—the "ADAPT Data Model." This translation allowed us to perform entity resolution, correct some problems in the CDM data we were receiving, and raise the level of abstraction to one more compatible with our analysis.

4.2.4 Engagement 4: Fully automated detection, cross-host analysis.

In the previous engagement, we produced detection results automatically. So for the five months leading up the Engagement 4, we worked with the TA5 team to define a method for evaluating the automatically produced real-time alarms using Splunk as the system of record. Splunk received all automated alarms from the ADAPT system and persists them unchanged for easy querying and future reference at any point. Despite the planning in the preceding months and subsequent agreement, the TA5 teams abandoned this evaluation method without notice, as described in section 4.1.

Applying the same objective evaluation methods used in Engagement 3, our team was able to classify all alarms as either being a false alarm, or a successful detection of one of the attack components. The following figure shows a section of the slide presented at the January 2019 PI meeting cataloguing the tallied results and correcting the erroneous draft report issued by the evaluation team (the final report was never published). The columns

near the right side of the chart show the results of the scripted evaluation the automated alarms ("Script") alongside the human judgment applied to those same alarms.



TA1 Team	Attack Name	Component Count	Detections Script	Detections Human	
fivedirections	1045_firefox_drakon	22	0	0	0%
fivedirections	1435_ssh_lwabeater	27	12	2	44%
fivedirections	1342_phishing_powershell	24	20	3	83%
fivedirections	0931_ssh_rdp_taken2	57	36	15	63%
fivedirections	1318_firefox_drakon	28	0	0	0%
fivedirections	1420_firefox_drakon	15	5	3	33%
trace	1351_vnc	10	8	6	80%
trace	1221_firefox_drakon	9	0	0	0%
trace	1505_metasploit	14	7	1	50%
trace	1313_firefox_drakon	19	8	7	42%
trace	0951_azazel	30	11	11	37%
marple	1538_ssh_attack	13	6	5	46%
marple	1047_firefox_drakon	16	0	0	0%
marple	1451_firefox_drakon	15	0	0	0%
marple	1556_metasploit	15	2	10	67%
marple	1403_ssh_attack	23	0	0	0%
cadets	1501_webshell	10	0	0	0%
cadets	1314_micro_apt	22	13	11	59%
cadets	1039_nginx_drakon	6	0	0	0%
cadets	1127_dropbear	15	0	0	0%
theia	1135_theia_dropbear	9	0	0	0%
theia	1543_theia_micro	11	3	3	27%
theia	1624_theia_pine_metasploit	12	3	9	75%
theia	1418_theia_firefox_drakon	26	8	2	31%
theia	1249_theia_firefox_drakon	8	0	0	0%
clearscope	1354_clearscope_microapt	9	0	0	0%
clearscope	1256_clearscope_ransomware	5	*	4	80%
clearscope	1310_clearscope_gatherapp	5	*	2	40%
marple-trace	1046_marple_trace_ssh	57	26	17	46%
marple-trace	1501_marple_trace_firefox_drakon	18	5	4	28%
fivedirections-clearscope	1644_5d_clearscope_ransomware	7	5	4	71%
fivedirections-clearscope	1550_5d_clearscope_firefox_drakon	7	0	0	0%
fivedirections-clearscope	1335_5d_clearscope_firefox_drakon	42	22	20	52%
cadets-theia	1013_cadets_theia_ssh_drakon	22	12	8	55%

Figure 8: Engagement 4 Attacks Detected by the ADAPT System
A section of a slide presented at the January 2019 PI meeting showing the attacks detected by the ADAPT system (green rows) using both fully automated and human detection. More attack components have successful automated alarms (the "Script" column) than successful human interpretation of those alarms (the "Human" column).

The reduced "Human" detections compared to the "Script" detections are illustrative of a remaining difficulty in aligning human understanding with the capabilities achieved by automated methods. We briefly reflect on this still-outstanding problem in the final section of this report.

4.2.5 Engagement 5: Support 30-times the previous data load, operating in a linearly scalable fashion with low resource usage.

Through the previous Engagement, the ADAPT system was designed for *future* efficiency and scalability, but without that efficiency represented in the implemented system. In the time leading up to Engagement 5, the ADAPT team focused on scaling the system to handle the 30-fold increase in data volume compared to the previous engagement, and integrating the Quine system to complete the full design of an efficient automatic APT detection system.

Since system performance was a major goal for this engagement, we describe it more in detail in the next section: 4.3.

Ground truth for the attacks performed in Engagement 5 was never released, so we were unable to do a comparable analysis for the detection results of the ADAPT system in Engagement 5. The TA5 team did release a PDF narrative of their attacks which was sufficient for the ADAPT team to test whether the automated alarms included some representation of the attacks described, even if not a full analysis of true and false alarms. The attacks known to be detected are listed in the following figure, shown at the final PI meeting:

Host	Attack Name	Alarm reported?
fivedirections-2	Firefox Drakon APT Elevate Copykatz Sysinfo	Yes
marple-1	Firefox Drakon APT	Yes
theia-1	Nmap SSH SCP	Yes
fivedirections-3	Nmap SSH SCP	Yes
fivedirections-2	Firefox BITS Micro APT	Yes
theia-1	Firefox Drakon APT BinFmt-Elevate Inject	Yes
clearscope-2	Appstarter APK Micro APT	Yes

Table 15: Known Detected Engagement 5 Attacks

4.3 System Performance

The primary difference between the final two engagements was a 30-fold data scale increase. The experimental methods used in Engagement 4 were tested using the machines available and requiring sometime more than 100 GB to run in the smaller data sizes. So the final challenge posed to our team was to scale the methods we were using to accommodate a massive increase in data size—and correspondingly a necessary increase in our analysis rate.

Leading up to Engagement 5, we replaced the Neo4j database with the Quine database to allow the system as a whole to scale to the capabilities of the Quine system. Since the entire system is a back-pressured, asynchronous actor system—where all communication is handled through message-passing—the ingest pipelines would scale automatically to the slowest component. The slowest component has always been the database;

consequently, our focus was largely centered on making this component scale to handle the new data volume.

We successfully reconfigured the system to use Quine as the database backend and were pleasantly surprised to find that in most cases, the processing bottleneck was the delivery of CDM data through Kafka. This was confirmation that the Quine system allowed the ADAPT system as a whole to scale well-beyond the necessary volume of data processing required for Engagement 5.

Toward the end of the first week of Engagement 5, the TA3 team experienced a network outage. This caused our clustered system to become disconnected and cease coordination and processing. We hadn't engineered for resiliency under these conditions, so we decided to restart ingest from the beginning of the data stream. Since our system runs entirely automatically, this was a trivial exercise and required no additional effort beyond restarting the application. Interestingly, this restart provided us with the chance to test the full capabilities of our system.

While reingesting the first week's data—where it is all available immediately, instead of being produced only as fast as TA1 teams can publish it—our system was able to run at full speed. During this "catch up" period, we observed the full ADAPT system ingest at a rate often as high as 134,000 CDM statements per second, while *simultaneously* running approximately 2,500,000 complex read queries. Even during this time, most of our ingesting systems were limited in their processing speed by the Kafka configuration

5.0 CONCLUSIONS

The structure of the Transparent Computing program did not lend itself well to producing scientifically valid results for APT detection. The ad hoc nature of the adversarial engagements provided the opportunity to simulate a controlled version of a real-world environment, however the control of that situation (e.g. scripted background activity, whitelisting administrator's APT-like activity) ended up making it quite artificial in the end. As a result, the scientific results we do have to share are scoped largely to individual methods developed along the way. Indeed, the creation of Categorical Anomaly Detection and advancement of the Quine distributed graph interpreter/database are substantial scientific advances. In this concluding section, we reflect on these core achievements and consider contributions of this program to future research.

5.1 Archiving the Transparent Computing Data and Making it Available for Future Research

The University of Edinburgh Data Library has a great deal of expertise archiving research data for reuse by other researchers. We view the datasets constructed in the TC Adversarial Engagement exercises (as well as derived datasets such as the ADM data or contexts) as a very valuable computer security research data resource both for our own future research and for the broader security and data management community. Although some of the data from Engagement 3 is already publicly available, it is not clear whether this will remain available over a longer-term. We have discussed this with University and DARPA staff and put them in contact with each other so that when other TC-related datasets are cleared for public release, the University will be ready to host them.

Besides simply hosting the raw CDM data, it is important to provide adequate documentation to allow researchers from outside the TC program to understand and make use of the data. This should include documentation of CDM itself, high-level descriptions of the different datasets, scenarios, and providers, and summaries of the "ground truth" information about the different attacks. We are already aware of colleagues in security working on similar techniques to those explored in TC who have tried to work with E3 data, but with limited success. Ensuring that the data is not just preserved but adequately documented could greatly increase the impact of the TC project on the broader computer security community.

5.2 Real-Time Graph Data Management

The problem of APT detection involves managing and analyzing many separate complex domains related to running systems: file system hierarchies, process parentage trees, network connections, memory references, user and group accounts, and many other operating system and program level abstractions. To understand what a computer is actually doing requires analyzing the interactions between these domains and understanding the significance of every connection. This makes graph databases a somewhat obvious, and arguably inevitable choice for a management system.

Over the course of this program, we used three different graph databases to manage the Transparent Computing data provided by other team members: Titan, Neo4j, and Quine. Neo4j and Titan represented the state-of-the-art as the commercial and open source leaders in the graph database space, however, our experience on this program underscored the shortcomings of these graph database systems.

The original design of the ADAPT system in the 2015 proposal took for granted that we would be able to ingest data, read it out for analysis, and write annotations back in—all at a rate sufficient to keep up with the data volume produced by TA1 teams. Once the TA1 teams began producing that data, it quickly became apparent that the volume of data for ingest alone was too high for existing graph databases to manage. The additional analysis and annotation tasks increase the burden on the database even further.

Quine is a prototype graph database developed by our team outside of the Transparent Computing program. It was designed to be run on a cluster of computers (of any size), with a single logical graph spread across the set of participating machines. As each member of the cluster manages a part of the graph, the cluster system as a whole is able to scale linearly to manage arbitrarily large amounts of data.

Quine also has the ability to process complex graph queries as "standing queries", which remain resident in the database and are process with an extremely efficient partial-evaluation strategy. This allowed the analysis and annotation tasks to be set on the database and left to resolve themselves as new data is streamed in. Combined with Quine's ability to ingest data on a single machine an order of magnitude faster than existing commercial databases, and linear scalability across a cluster, these features made Quine the perfect choice for the graph data management problem. Combined with a streaming, back-pressured ingest system, Quine was able to easily handle the tremendous volume of data from Engagement 5 and will scale well beyond.

5.3 Finding Novel Behavior

APT detection is a needle-in-a-haystack problem. High volumes of typical behavior data vastly drown out the lone signal we hope to pick up on. Because one small event might indicate the activity betraying the existing APT, all activity has to be managed and analyzed.

The technique for Categorical Anomaly Detection using Prediction by Partial Matching described above proved to be a very effective method for:

- efficiently learning a probabilistic model of system behavior (a baseline).
- analyzing all activity in real-time, even high volumes of data.
- calculating the novelty of each observed action.
- generating an alarm with relevant context for sufficiently novel activity.

Combined with the strengths of the Quine distributed graph database/interpreter, these capabilities are practical and needed in current enterprise security systems.

5.4 Open Questions

In the most difficult APT example, an advanced adversary can use a zero-day exploit to inject code into a running process and simply corrupt existing data. If the corrupted process normally accesses that data, there is nothing in this scenario that is novel, anomalous, suspicious, or even detectable. —and yet, it was malicious.

Is some normal behavior malicious?

So while some of the methods described above—particularly Categorical Anomaly Detection—proved to be very effective for finding the needle-in-the-haystack, we believe that there still remain some threats that cannot be detected. —certain adversaries will leave behind only a needle-less pile of hay.

While this Categorical Anomaly Detection technique that we developed on this program was very effective, and produced low numbers of false alarms after learning a sufficient baseline, for every true alarm produced by the system we are still left with the final problem addressed to a human:

Is anomalous behavior malicious?

This is the question that still hangs over all of the research on this program. Hunting for a "threat" implies that we know what constitutes a threat. Despite the progress that we have made in automated detection, this final judgement requires human attention. Perhaps this reflects only our reticence to let a computer make the final judgment; or maybe the problem is underspecified, and before we can make fully automated advanced persistent thread detectors, we have to learn for ourselves what we mean by "threat."

6.0 REFERENCES

[Hewitt et al., 1973] Hewitt, Carl, Peter Bishop, and Richard Steiger. "A universal modular actor formalism for artificial intelligence." Proceedings of the 3rd international joint conference on Artificial intelligence. Morgan Kaufmann Publishers Inc., 1973.

[Siddiqui et al., 2015] Md Amran Siddiqui, Alan Fern, Thomas Dietterich, and Weng-Keen Wong. (2015). Sequential Feature Explanations for Anomaly Detection. SIGKDD Workshop on Outlier Detection and Description.

[Siddiqui et al., 2019] Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, and Weng-Keen Wong. (2019). Sequential Feature Explanations for Anomaly Detection. ACM Transactions on Knowledge Discovery from Data, 13 (1).

[Siddiqui et al., 2018a] Md Amran Siddiqui, Alan Fern, Thomas Dietterich, Ryan Wright, Alec Theriault, and David Archer. (2018). Feedback-Guided Anomaly Discovery via Online Optimization. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-2018).

[Das et al., 2016] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, and Alan Fern. (2016). Incorporating Expert Feedback into Active Anomaly Discovery. IEEE International Conference on Data Mining (ICDM-2016).

[Siddiqui et al., 2016] Md Amran Siddiqui, Alan Fern, Thomas Dietterich, and Shubhomoy Das. (2016). Finite Sample Complexity of Rare Pattern Anomaly Detection. Conference on Uncertainty in Artificial Intelligence (UAI-2016).

[Siddiqui et al., 2018b] Md Amran Siddiqui, Alan Fern, Ryan Wright, Alec Theriault, Dave Archer, William Maxwell. (2018). Detecting Cyberattack Entities from Audit Data via Multi-View Anomaly Detection with Feedback. AAAI Workshop on Artificial Intelligence in Cyber Security (AICS).

[Das et al., 2017] Shubhomoy Das, Weng-Keen Wong, Alan Fern, Thomas Dietterich and Md. Amran Siddiqui. (2017). Incorporating Feedback into Tree-based Anomaly Detection. KDD Workshop on Interactive Data Exploration and Analytics.

[Liu et al., 2008] Fei Liu, Ting Kai Ming, and Zhi-Hua Zhou. (2008). Isolation Forest. International Conference on Data Mining.

[Chandola et al., 2010] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection for discrete sequences: A survey." IEEE transactions on knowledge and data engineering 24.5 (2010): 823-839.

[Abreu et al. 2016] Rui Abreu, Dave Archer, Erin Chapman, James Cheney, Hoda Eldardiry, Adrià Gascón: Provenance Segmentation. TaPP 2016 https://www.usenix.org/conference/tapp16/workshop-program/presentation/abreu

[Agrawal et al. 94] Rakesh Agrawal and Srikant Ramakrishnan: Fast Algorithms for Mining Association Rules in Large Databases VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases, Pages 487-499.

[Akoglu et al. 2012] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. 2012. Fast and reliable anomaly detection in categorical data. In Proceedings of the 21st ACM international conference on Information and knowledge management (CIKM '12), 415-424.

[Berrada and Cheney 2019] Ghita Berrada and James Cheney. Aggregating unsupervised provenance anomaly detectors. 11th International Workshop on Theory and Practice of Provenance (TaPP 2019)
https://www.usenix.org/conference/tapp2019/presentation/berrada

[Berrada et al. 2019] Ghita Berrada, Sidahmed Benabderrahmane, James Cheney, William Maxwell, Himan Mookherjee, Alec Theriault, and Ryan Wright. A baseline for unsupervised advanced persistent threat detection in system-level provenance. arXiv: https://arxiv.org/abs/1906.06940

[Chan et al. 2017] Sheung Chi Chan, Ashish Gehani, James Cheney, Ripduman Sohan, Hassaan Irshad: Expressiveness Benchmarking for System-Level Provenance. TaPP 2017. https://www.usenix.org/conference/tapp17/workshop-program/presentation/chan

[Ganter and Stumme 2003] Bernhard Ganter and Gerd Stumme. Formal Concept Analysis: Methods and Applications in Computer Science. TU Dresden, Technical Report, 2003

[Ganter and Wille 1997] Bernhard Ganter and Rudolf Wille. Formal Concept Analysis: Mathematical Foundations, 1st ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997.

[He et al. 2005] Zengyou He, Xiaofei Xu, Joshua Zhexue Huang, and Shengchun Deng. FP-outlier: Frequent pattern based outlier detection. Comput. Sci. Inf. Syst., 2(1):103–118, 2005.

[Koufakou et al. 2007] Anna Koufakou, Enrique G. Ortiz, Michael Georgiopoulos, Georgios C. Anagnostopoulos, and Kenneth M. Reynolds. A scalable and efficient outlier detection strategy for categorical data. In ICTAI 2007, pages 210–217, 2007.

[Lin 2010] S. Lin. Rank aggregation methods. WIREs Computational Statistics, 2(5):555–570, 2010.

[Mookherjee 2018] Evaluation of unsupervised anomaly detectors using operating system level application data. Himan Mookherjee. MSc thesis, University of Edinburgh, 2018.

[Narita and Kitagawa 2008] Kazuyo Narita and Hiroyuki Kitagawa. Outlier detection for transaction databases using association rules. In WAIM, pages 373–380, 2008.

[Pasquier et al. 2017] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David M. Eyers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In SoCC 2017.

[PROV] https://www.w3.org/TR/prov-dm/

[ProvMark] ProvMark repository. https://github.com/arthurscchan/ProvMark

[Smets and Vreeken 2011] Koen Smets and Jilles Vreeken. The odd one out: Identifying and characterising anomalies. In SDM 2011, pages 804–815, 2011.

[Szathmary et al. 12] Laszlo Szathmary, Petko Valtchev, Amedeo Napoli, and Robert Godin: Efficient Vertical Mining of Minimal Rare Itemsets. CLA - The Ninth International Conference on Concept Lattices and Their Applications - 2012, 2012, Fuengirola, Spain.

[Tan et al. 2013] Swee Chuan Tan, Si Hao Yip, Ashfaqur Rahman. One Pass Outlier Detection for Streaming Categorical Data. The 3rd International Workshop on Intelligent Data Analysis and Management. Springer Proceedings in Complexity, DOI: 10.1007/978-94-007-7293-9_4

[Vreeken et al. 2011] J. Vreeken, M. van Leeuwen, and A. Siebes. KRIMP: Mining itemsets that compress. Data Mining and Knowledge Discovery, 23(1):169–214, 2011.

[Wille 1992] Rudolf Wille. Concept lattices and conceptual knowledge systems. Computers & Mathematics with Applications, vol. 23, no. 6, pp. 493-515, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0898122192901207

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

ACRONYM DESCRIPTION
AD – Anomaly Detection
APT – Advanced Persistent Threat
AUC – Area Under Curve
AVF - Attribute Value Frequency
CDM - Common Data Model (format created on the TC program)
FCA - Formal Context Analysis
MFP - Minimum Feature Prefix
nDCG - Normalized Discounted Cumulative Gain
OCO - Online convex optimization
PAC - Probably Approximately Correct
PPM – Prediction by Partial Matching
ROC – Receiver Operator Characteristic
RPAD - Rare Pattern Anomaly Detection
SFE - Sequential Feature Explanation
TC - Transparent Computing (DARPA program)
UI - User Interface
UUID - Universally Unique Identifier