Professional Thesis Report

Shubham Rai

August 25, 2020

Behavioral Threat Detection: Detecting Living of Land Techniques

Company: ReaQta

Company Supervisor: Giuseppe Massaro

Academic supervisor: Marc Dacier

Non-Confidential

Digital Security Track

EURECOM

I hereby confirm that this report is my own work and I have documented all sources and material used.

Amsterdam, August 25, 2020

(Shubham Rai)

Abstract

Behavior analysis is an important component of any intelligent anomaly detection system, where anomalies are abnormal observations that violate a typical behavior or cause a sudden change in behavior.

Recent antivirus and machine learning-based malware detection have all increased their effectiveness in detecting file-based attacks, consequently adversaries have migrated to “living off the land” (Lotl) techniques to bypass the advanced security detection. The adversaries practice this by executing system tools preinstalled within the operating system or commonly brought in by administrators to carry out tasks like automating IT administrative tasks, running scripts for operations, executing code on remote systems, and much more such tasks, which goes undetected by even advanced signature detection based systems.

Currently there exists wide range of methods that try to detect anomalies from the system’s logs by analyzing and applying numerous algorithms such as heuristic based and machine learning algorithms. However, experts argue that security anomalies cannot be found by looking into a single event, rather looking into the relationship or chains of events is necessary to understand the root cause of a problem. One of the tried and tested approach against Lotl attacks that has been used by security researchers and security analysts is to use the MITRE ATT&CK framework for detecting such attacks by authoring detection rules against suspicious parent-child processes.

In this thesis work, we aim to automate the task of rule generations and propose a novel detection method that can effectively capture anomalous parent-child behavior of the system processes. The first task for this work would be to look at an event as parent-child relationship instead of looking at single events and finding anomalous patters therein, thus enabling us to analyze Lotl techniques used by attackers. The detection system is supposed to create rules and statistics for detection, based on the parent-child process relationships, which in other case is extremely hard to filter manually (writing rules) and no matter how effective is the detector, its logic can only solve one specific attack. A failure of detectors to generalize and detect emergent attacks presents a unique opportunity for machine learning, explored in this thesis.

Résumé

L'analyse du comportement est une composante importante de tout système intelligent de détection des anomalies, où les anomalies sont des observations anormales qui violent un comportement typique ou provoquent un changement soudain de comportement.

Les antivirus récents et la détection des logiciels malveillants basée sur l'apprentissage de la machine ont tous augmenté leur efficacité dans la détection des attaques basées sur des fichiers, par conséquent les adversaires ont migré vers les techniques de "vivre de la terre" (Lotl) pour contourner les détections de sécurité avancées. Les adversaires pratiquent cette technique en exécutant des outils système préinstallés dans le système d'exploitation ou couramment introduits par les administrateurs pour effectuer des tâches telles que l'automatisation des tâches administratives informatiques, l'exécution de scripts pour les opérations, l'exécution de code sur des systèmes distants, et bien d'autres tâches de ce type, qui ne sont pas détectées même par les systèmes avancés basés sur la détection de signatures.

Il existe actuellement un large éventail de méthodes qui tentent de détecter les anomalies des journaux du système en analysant et en appliquant de nombreux algorithmes tels que les algorithmes basés sur l'heuristique et l'apprentissage machine. Cependant, les experts affirment que les anomalies de sécurité ne peuvent pas être trouvées en examinant un seul événement, mais qu'il faut plutôt examiner la relation ou les chaînes d'événements pour comprendre la cause première d'un problème. L'une des approches éprouvées contre les attaques Lotl qui a été utilisée par les chercheurs et les analystes en sécurité consiste à utiliser le cadre MITRE ATT&CK pour détecter de telles attaques en créant des règles de détection contre les processus parents-enfants suspects.

Dans ce travail de thèse, nous visons à automatiser la tâche de génération des règles et proposons une nouvelle méthode de détection qui peut efficacement capturer le comportement anormal parent-enfant des processus du système. La première tâche de ce travail serait de considérer un événement comme une relation parent-enfant au lieu d'examiner des événements uniques et d'y trouver des modèles anormaux, ce qui nous permettrait d'analyser les techniques Lotl utilisées par les attaquants. Le système de détection est censé créer des règles et des statistiques de détection, basées sur les relations de processus parent-enfant, qui dans d'autres cas sont extrêmement difficiles à filtrer manuellement (écriture de règles) et quelle que soit l'efficacité du détecteur, sa logique ne peut résoudre qu'une attaque spécifique. L'incapacité des

détecteurs à généraliser et à détecter les attaques émergentes présente une opportunité unique pour l'apprentissage machine, explorée dans cette thèse.

Acknowledgements

First and foremost, I would like to thank my supervisor Mr. Marc Dacier, for his continued periodic feedbacks, advices, and suggestions throughout the thesis. The support has been instrumental in shaping up the thesis related work.

I would also like to take this opportunity to thanks Alberto Pelliccione for making this thesis possible and giving an opportunity to work or such an interesting topic. Further, goes without saying, thanks to Giuseppe Massaro, under whose guidance, I have been able to complete the thesis. Throughout the bottlenecks that I had, Giuseppe has helped me a lot in overcoming these, brainstorming, and coming up with ideas for the problems. Also, I would like to extend my gratitude towards everyone at ReaQta, for being so helpful and friendly to make this time memorable.

Finally, the times have been very uncertain everywhere due to the pandemic, so I would personally thank everyone who has supported me during this period.

Contents

Abstract **3**

Résumé **4**

Acknowledgements **6**

Chapter 1 **9**

Introduction 9

1.1 Introduction 9

1.2 Motivation 10

1.3 Research Questions 11

Chapter 2 **13**

Theoretical Background 13

2.1 Living off the Land Attacks 13

2.1.1 Dual Nature of Tool 14

2.2 What are Anomalies? 15

2.3 Intrusion Detection System and Categorizations 17

2.3.1 Signature-based intrusion detection systems (SIDS) 18

2.3.2 Anomaly-based intrusion detection system (AIDS) 18

2.3.3 Implementation Techniques for Anomaly Based Intrusion Detection System (AIDS) 19

Chapter 3 **24**

Literature Survey and Related Work 24

3.1 Earliest Works - IDS 24

3.2 Rule Mining in Anomaly Detection 25

3.3 Dynamic Graph based Detection 26

3.3.1 Anomaly Detection in Time Evolving Graphs 26

Chapter 4 **29**

Methodology and Proposed Work 29

Approach 1: Rule Learning based Approach 29

4.1 Data Mining and Association Rule Mining 29

4.2 Learning rules for Anomaly detection algorithm (LERAD) 31

4.2.1 Rule Learning 32

4.2.2 Candidate rule generation 32

4.2.3 Removing duplicate rules 33

Approach 2: Dynamic Graph based Approach 34

4.3 Graphs and why graphs? 34

4.4 Streaming Data Structures 35

4.5 Count-Min Sketch (CMS) 36
4.6 Data Model 37
4.6.1 Edge Scoring 37
4.6.2 Laplacian Feature Scoring 39
4.6.3 Community detection and Scoring 40
Chapter 5 **42**
System Setup and Methodology 42
5.1 General Setup and Data Source 42
5.2 Data Engineering and Preprocessing 44
5.3 Dataset Generation 46
Chapter 6 **49**
Evaluation and Outcome 49
6.1 Experiment: LERAD 49
6.2 Experiment: Statistical Model 50
Chapter 7 **54**
Conclusion 54
7.1 Attacks against the Detection Service 54
7.2 Summary 55
References **56**

Chapter 1

Introduction

1.1 Introduction

The ever-lasting challenge of detecting and mitigating security anomalies in computer systems has become more so essential than ever, especially with the enormous number of connected devices and growing sophistication of cyber-attacks.

According to the Symantec Internet Security Threat Report from 2019, zero-day exploit usage declined, only 23 percent of attack groups were known to use zero days, down from 27 percent in 2017. Also, attacks which rely solely on "living off the land" (Lotl) techniques and do not use any malicious code were on a rise. The targeted attack group "Gallmaker" is an example of this shift, with the group exclusively using generally available tools to carry out its malicious activities [1]. Supply chain attacks and Lotl based attacks continued to be a feature of the threat landscape, with attacks increasing by 78 percent in 2018 [1]. As pointed out in the 2017 Data Breach Statistics, roughly nine billion data records were lost/ stolen by hackers since 2013 (Breach Leve Index, 2017 [2]. Historically, cybercriminals have primarily focused on bank customers, stealing bank accounts, or stealing credit card information, however, the new generation malwares have become more advanced and ambitious and are targeting the infrastructure such as banks themselves, often trying to take millions of dollars in one attack.

There is one emerging trend in most of the recent cyberattacks, including the famous Petya/ NotPetya in 2017, that they were not carried out by installing malware; rather, they used supposedly secure programs. According to a report by X-Force in 2018 more than half (57 percent) of attacks analyzed by X-Force in 2018 [3] did not leverage installing a malware and many involved the use of non-malicious tools including PowerShell and PsExec to evade detection systems. These so-called dual-use tools provide threat actors with the ultimate playground, where they know it is easier to hide inside approved softwares and services than it is to drop new, malicious software in another's environment.

For that reason, the detection of unseen malwares and zero-day attacks has grown to be one of the top security priorities for researchers. High profile incidents of cybercrime show the ease with which cyber threats can spread globally and disrupt business services and facilities. Cybercriminals around the world are motivated to steal information, illegitimately receive monetary benefits, and acquire new targets. Malware is intentionally created to compromise computer systems and take advantage of any weakness in intrusion detection systems, which makes it even necessary to develop efficient IDS to detect novel, sophisticated malwares.

Analyzing and monitoring system logs for threats and intrusion is not a new trend and has been historically one of the most researched fields for intrusion detection. System event logs is a rich source for detecting anomalies, but the scale and inconsistent formatting of the events logs are big issues to tackle, but otherwise it can be good way of detecting anomalies across platforms. The value from this thesis would be towards using a rule mining and streaming data/ graph-based technique towards anomaly detection that would integrate machine learning technique to heuristic and statistics-based anomaly detection.

1.2 Motivation

The work in this thesis was carried out as part of my internship in ReaQta, and this work on behavioral anomaly detection holds both business and academic value, since the company works in the field of Advanced Persistent Threat (APT) detection.

There are several reasons why creating a behavioral anomaly detection system based on logs could be an effective system, especially to detect attacks using “Living off the land” (Lotl) techniques. Not only properties like unseen log patterns, absence of certain logs, high/ low frequency of logs etc. can indicate potential anomalies but logs-based anomaly detection can also help in recreating the incidence and help in post infection analysis. Since the number of logs that are generated from modern IT infrastructure is huge finding security anomalies can be a tedious task as its equivalent to finding needle in haystack.

Another problem is the evolution of threats and attacks, and that it is getting harder to detect, so it could be difficult to find out if system events are legitimate or malicious. Anomaly detection systems are doing a decent job in detecting malicious traffic, but a standard detection system must continuously be updated with rulesets or signatures and upgrades should be up to date with the recent threat vectors. Big companies like Symantec, Norton etc. releases new rulesets on a regular basis, but even these might not be sufficient, considering the vast space of these rules. How to keep up with the ever-increasing rulesets and ensure that these rules are effective enough, is something that many security researchers have inquired.

The objective is not just to create an anomaly detection system but also come up with an anomaly scoring mechanism, that adds risk/ threat score to the anomalies, which adds business value as well. The motivation also comes from using the existing backend services used by ReaQta to generate threats and behavioral process trees for security analysts. Hence, we shall aim at detecting attacks that use "Living off the land techniques" (Lotl), where the attackers use the existing tools used by system administrators to exploit system. Attackers can use trusted OS tools like powershell.exe, wmic.exe, or schtasks.exe, which can be difficult to differentiate with normal usage of these tools. To target this, security researchers author rules for suspicious parent child relationships and we shall be defining anomalies as anomalous parent-child events in this thesis work.

Many researchers have been using Machine Learning for anomaly detection, and we shall be using it as well but look at the anomalies from a parent child process perspective. Since manually doing this is a tedious and time-consuming task, we shall be using two approaches – one based on association rule learning and one based on statistics on parent child relationships and evaluate the results achieved from the same.

1.3 Research Questions

The motivation mentioned in the earlier section has led us to clearly define the following research questions and we shall be trying to find answers to these in this thesis work.

RQ1: Can we establish parent child relationship in event logs and detect anomalies based on that?

RQ2: Can we use rule mining approach to build rulesets that can efficiently detect parent-child anomalies?

RQ3: Since parent-child relationships can be expressed efficiently in graphs, can we use data streams and graph-based approach to detect parent-child anomalies?

RQ4: What can be possible weaknesses and exploits in the mentioned approaches?

RQ5: What are the possible scenarios and use cases where the mentioned approach can be adopted?

First three questions shall be answered as the outcome of this thesis, where we would we exploring 2 techniques for detection anomalies based on parent-child relationship. We will be generating the parent-child transformations from the events generated by system logs and apply rule mining and graph-based approach on the transformation, which will be covered in the Chapter 4, 5, 6, and answer RQ1, RQ2 and RQ3. The goal will also be to measure the efficiency and performance of these two approaches and their applicability to real world dataset. RQ4 and RQ5 would be answered in the summary section of the thesis.

We will be looking in detail at the parameters used during our experiments and the rationale for deciding the tuning parameter for both detection approach. We shall be also elaborating on the relationships and features used to build the models used in evaluating the dataset and based on the approach used we will look at the possible attacks possible on our approach used. The last question will be answered based on the dataset we used and result outcome, and how applicable is parent-child relationship in outlier detection.

Chapter 2

Theoretical Background

In this Chapter we shall be presenting some necessary background information on anomaly detection, in general and on necessary related background that will be used throughout this thesis. A description of the different research fields, on which we have focused, will also be presented as necessary relevant related work.

2.1 Living off the Land Attacks

The use of “Living off the Land” (Lotl) tactics and tools by cyber criminals has been a growing trend within the cyber security landscape in recent times. The concept of Lotl technique is not a novel approach and has been around for as long as 25 years. Using system tools as backdoors has been a common practice in past, however, in recent years, it has returned and grown in importance.

In the technology world, Lotl refers to an attacker’s behavior that uses tools or features that already exist in the target environment. Such attackers exploit ways to fly under the radar of either intrusion prevention or detection technologies. Generally, hash values/ other indicators of comprise (IOCs) are used by signature-based intrusion prevention technologies to detect and quarantine/ isolate malicious processes. Lotl attacks thus get onto an organizations’ systems via trusted programs/ software’s that aren’t going to flag any suspicions, then inject them with malicious payload. There are multiple advantages of using this tactic: to begin with, they are able to get around traditional protection systems, which will not be triggered by unusual use of apparently secure software. Given the circumstances, it is much more difficult to identify where the attack comes from compared to when certain files are being used. The reason being that most cybersecurity solutions are unable to detect dangerous behavior when it is carried out using tools classified as legitimate [4].

While attackers can change IOCs relatively easily, refer Figure 1 for the Pyramid of Pain, using pre-existing software avoids the process being flagged as suspicious and saves the attacker cycles in developing a binary from scratch to deliver an attack [5].

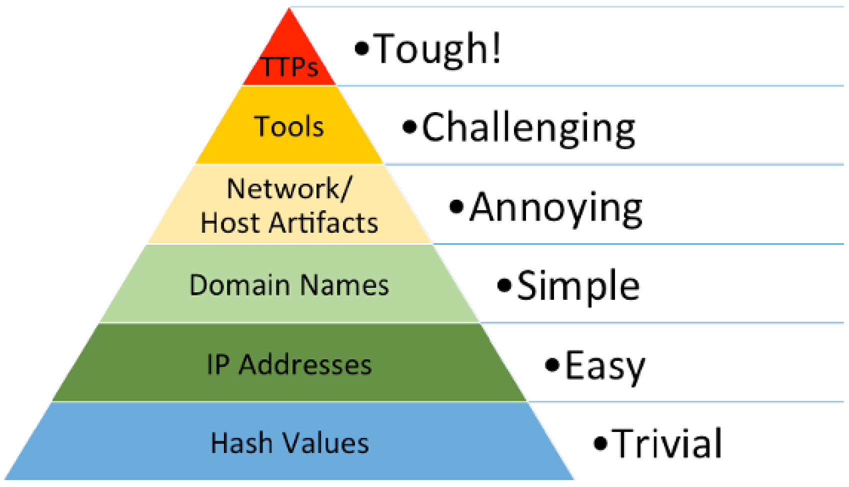
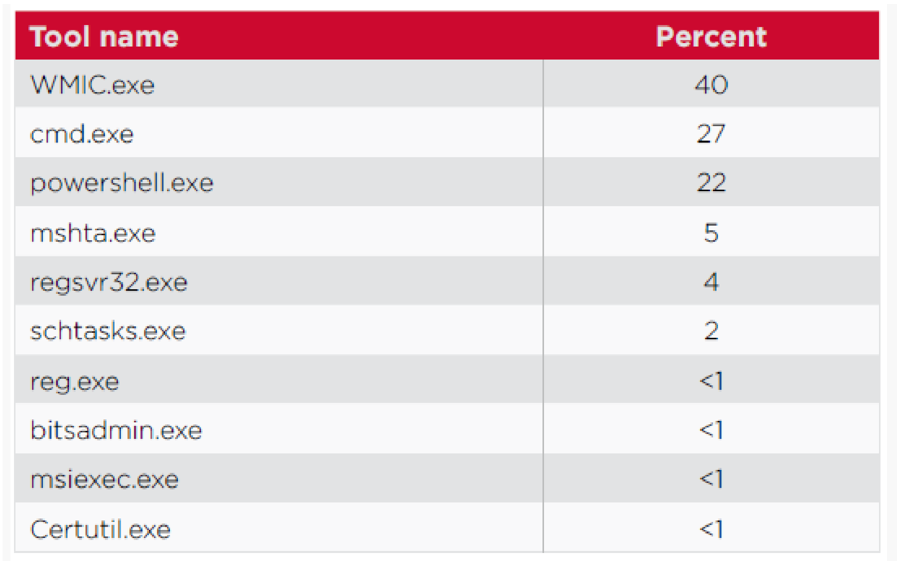


Figure 1: The pyramid of pain represents the difficulty levels for attackers to change indicators that security analysts may use to detect their activity

2.1.1 Dual Nature of Tool

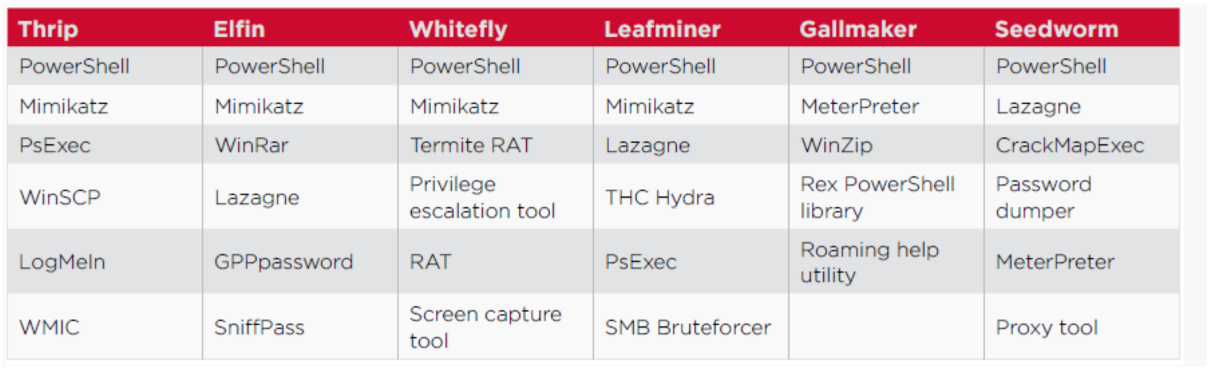
As observed by Symantec in Q1 2019 [4], the most frequently executed dual use tools observed were net.exe, PowerShell, the certification utility, the task scheduler, and the WMI command line (WMIC), however, the percentage usage in malicious activities was significantly low. The context of the event and execution pattern remains one of the key components in detecting behavior of such attacks and this is what we try to achieve with machine learning and statistics in this thesis. The same report by Symantec also shows that WMI, the command line tool, and PowerShell were amongst the most frequently used for malicious purposes, accounting for 89 percent of all dual-use tools used as downloaders, as also shown in the Figure 2 below.



Tool name	Percent
WMIC.exe	40
cmd.exe	27
powershell.exe	22
mshta.exe	5
regsvr32.exe	4
schtasks.exe	2
reg.exe	<1
bitsadmin.exe	<1
msiexec.exe	<1
Certutil.exe	<1

Figure 2: Dual Use tools reported by Symantec [4].

We also look at the target attack groups, which have been known to using Lotl tactics for a long time, with almost all active groups were seen using dual-use tools at some point. Looking at the stats from the six active target groups as in Figure 3, we can observe that PowerShell is the most common dual-use tool, utilized in about 77% of the target group attacks [4].



Thrip	Elfin	Whitefly	Leafminer	Gallmaker	Seedworm
PowerShell	PowerShell	PowerShell	PowerShell	PowerShell	PowerShell
Mimikatz	Mimikatz	Mimikatz	Mimikatz	MeterPreter	Lazagne
PsExec	WinRar	Termite RAT	Lazagne	WinZip	CrackMapExec
WinSCP	Lazagne	Privilege escalation tool	THC Hydra	Rex PowerShell library	Password dumper
LogMeIn	GPPpassword	RAT	PsExec	Roaming help utility	MeterPreter
WMIC	SniffPass	Screen capture tool	SMB Bruteforcer		Proxy tool

Figure 3: Dual-use tool usage among active targeted attack groups [4].

2.2 What are Anomalies?

Anomalies can be described as patterns in observations/ dataset that do not conform to a well-defined notion of normal behavior [6]. Figure 4 shows anomalies in a simple 2-dimensional data set. The data has two normal regions, N_1 and N_2 since most observations lie in these two regions. Points that are sufficiently far away from the regions, e.g., points o_1 and o_2, and points in region o_3, are anomalies/ outliers.

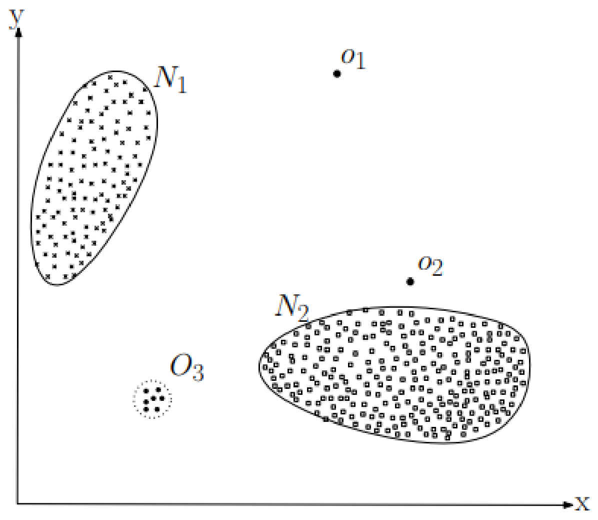


Figure 4: Example showing anomalies in a 2D dataset [6].

Anomalies can be introduced in the data due to a variety of reasons, such as malicious activity, e.g., CC fraud, cyber-intrusion, terrorist activity or system mal-functioning, but all these reasons still share common characteristics in the global data, which is what is of interest to the security analyst. Anomalies are however different from noise or novelty in data, where the former is the objective to reduce in data analysis and the latter is basically appearance of new patterns that would be later incorporated into the normal characteristic model.

A necessary consideration of an anomaly detection technique is the nature of the desired anomaly. Anomalies, usually can be divided into following three main categories [6]:

- *Point Anomalies* – This is probably the simplest form of anomaly and focus of many security researchers. When an individual data element can be considered as outlier with respect to the rest of data, then the instance is termed as a point anomaly.
  For example, consider anomaly detection in CPU usage. Let the data set correspond to the CPU utilization of a system under observation. For the sake of simplicity, let us assume that the data is defined using only one feature - memory usage (univariate distribution). A transaction for which the memory usage is very high compared to the normal range of for a process will be a point anomaly.

- *Contextual Anomalies* - When a data element is anomalous in a given context, but not in general, then it is termed as a contextual anomaly (also referred to as conditional anomaly [7]).

  Using the information of the behavioral attributes in a specific context, the anomalous behavior is defined. A data instance, in specific context, might be a contextual anomaly, but similar data observation (in behavioral attributes) might be deemed normal in a different context. This property is important in identifying contextual and behavioral attributes for a contextual anomaly detection technique

- Collective Anomalies. When a collection of related data elements appears anomalous with respect to the complete data set, then it is can termed as a collective anomaly. These individual data instances in a collective anomaly scenario may not be anomalies individually, but their occurrence together as a collection is anomalous.

2.3 Intrusion Detection System and Categorizations

The main challenge in sophisticated malicious attacks is to identify the novel and obfuscated intrusion techniques, as the malware authors often use techniques such as information concealing, and evasion etc. to hide from detection by an IDS. This indeed calls for more sophisticated anomaly detection systems that can detect unauthorized intrusions at an early stage. Intrusion detection [8] and fraud detection [9] are one of the most researched fields within anomaly detection, however, we shall focus on intrusion detection domain in this thesis work. Algorithms from different domains maybe suitable for our purpose with some adjustments, for example in intrusion detection it is required to block anomalies real time, while in fraud detection there is usually a time interval to take action, but accuracy is of utmost importance so as to not miss a fraud.

Intrusion can originate from both within and outside the network, cause data breach, hamper the availability of IT systems, and cause monetary losses. IDSs aim at detecting intrusions and alerting system analysts at an early stage before the entire system gets compromised. In general, IDS systems can be classified into following 2 groups based on detection method used:

- *Signature-based Intrusion Detection System (SIDS), and*
- *Anomaly-based Intrusion Detection System (AIDS).*

2.3.1 Signature-based intrusion detection systems (SIDS)

Signature intrusion detection systems (SIDS) analyze data for signatures that match known cyberattacks using pattern matching techniques to detect known intrusions. Basically, when signature of an intrusion matches with the signature of a previous intrusion that already exists in the signature database, an alarm is flagged. For SIDS, host's logs are inspected to find sequences of commands or actions which have previously been identified as malicious or malware.

In the literature, SIDS are also referred as Knowledge-Based Detection or Misuse Detection [10]. The main idea is to build a database of intrusion signatures (from past or known occurrences) and to compare the current activities against the existing signatures and raise an alarm in case a match is found. While SIDS is effective in detecting known malware attacks [11], it does provide much protection against novel attacks or polymorphic variants of malwares.

2.3.2 Anomaly-based intrusion detection system (AIDS)

AIDS has drawn interest from a lot of scholars due to its ability to overcome the limitation of SIDS. A normal model of the behavior of a computer system is built in AIDS over time using statistical-based, machine learning or knowledge-based methods. Any significant deviation between the observed behavior and the model can regarded as an anomaly, which can then be interpreted as an intrusion. The assumption for this group of techniques is that malicious behavior differs from a normal user behavior.

The behavior of a user which is dissimilar to standard behavior is classified as intrusions. AIDS development comprises of two phases: the training phase and the testing phase. In the training phase, the normal profile is used to learn or create a model of normal behavior, and then in the testing phase, a new data set is used to establish the system's ability to generalize to previously unseen intrusions. Based on the method used for training, AIDS can be classified into several categories, for example, statistical based, knowledge-based and machine learning based [12]. The main advantage of AIDS is its ability to identify zero-day and novel attacks, since recognizing the abnormal user activity does not rely on a signature database.

Further, IDS can be categorized based on the dataset used, broadly into 2 categories, namely, host-based, network-based (router-based is also part of network-based IDS, since both are based on similar datasets) [13].

Host-based IDS (HIDS) is usually deployed on host-machines to monitor activities on that host machines. Earlier HIDS implementation used to take away much of the computing power from the users of the host machines, but now with host machines being more powerful, host-based intrusions systems have become extremely powerful. HIDS also provides the functionality of sandboxing/ isolating the compromised hosts, that can prevent the spread and administrators can accordingly handle the threats. HIDS can provide protection against both insiders and outsiders.

Network-based IDSs (NIDS) are usually installed on some strategic computers (generally the ones on the network periphery) in the network to monitor data packets sent between host machines. With NIDS, since network traffic data volume is so huge it presents difficulty in efficiently processing data and have limited ability to inspect all data in a high bandwidth network [14].

In my internship project, we have been working on a host-based anomaly detection system, which works on the event information captured from the hosts. The IDS is light weight and does not hamper the performance of the system, while the anomaly analysis is performed at the central server running the service. More on the details of the systems will be covered in Chapter 5.

2.3.3 Implementation Techniques for Anomaly Based Intrusion Detection System (AIDS)

AIDS methods can be classified into three main groups [15][16]:

- *Statistics-based*
- *Knowledge based*
- *Machine learning-based*

Statistics-based techniques

A statistics-based IDS builds a distribution model for normal user behavior profile, then detects low probability events and flags them as potential intrusions. Statistical AIDS often utilizes the statistical metrics such as the median, mean, mode and standard deviation of dataset samples. Statistical AIDS can identify any type behavioral differences from the given normal behavior. In general, statistical IDS normally use one of the following models:

- *Univariate Model*

"Uni" aka "one", implies that data source has only one attribute or variable. This technique is widely used when a statistical normal profile is built for a single measure of behavior in IT systems. Univariate IDS look for irregularities in each individual metric [17].

- *Multivariate Model*

It is based on relationships among two or more variables to understand the relationships between variables. The model can yield better classification, achieved from combinations of correlated measures rather than analyzing them separately. In [17] the author examines a detection system using multivariate quality control method by creating a long-term model of normal activities. The main challenge for a multivariate statistical IDS is that it is hard to approximate distributions for high-dimensional data.

- *Time-Series Model*

The observations are made over a certain time interval and a new observation is flagged abnormal if the probability of its occurrence at that time interval is too low. In [18] authors used time series for processing intrusion detection alert aggregates. In [19] authors presented a method for detecting network abnormalities by observing the abrupt variations found in time series data.

Knowledge-based techniques

Also, referred to as an expert system method, this technique-based approach requires creating a knowledge base which indicates the normal user profile. Actions which differ from this standard profile are flagged as an intrusion. Unlike in previous other classes of AIDS, the standard profile model is usually built by security analysts aka based on human knowledge, in terms of defining a set of rules that describe normal system activity.

An inherent benefit of knowledge-based techniques is the capability to reduce false-positives since the system has knowledge about all the normal behaviors. However, in dynamic environments, these type of IDS needs a regular update on knowledge for the expected normal behavior which is a time-consuming task. Knowledge-based techniques are further categorized in 3:

- *Finite State Machine*

FSM is a computation model that is used to represent and control execution flow. In general, the model is represented in the form of states, transitions, and activities, where a state checks the history data.

- *Description Language*

Description language is used to describe the syntax of rules which may be used to define the characteristics of a specific attack. These rules could be built by description languages such as N-grammars and UML [20].

- *Signature analysis:*

This is one of the earliest techniques applied in IDS, it relies on the simple idea of string matching, where an incoming packet is inspected, word by word, with a distinct signature and if a signature is matched, an alert is flagged.

AIDS based on machine learning techniques

- *Supervised Machine Learning for Anomaly Detection*

In supervised learning method a labeled training dataset with normal and anomalous samples is required for constructing a predictive model. The most common supervised methods include decision trees, rule-based systems, neural networks, support vector machines, naive Bayes and nearest-neighbor.

Arguably, the most popular non-parametric technique in supervised learning is K-nearest neighbor (k-NN) that calculates an approximate distances between different points on the input vectors and assigns the point to the class of its K-nearest neighbors. Another effective model is the Bayesian network that encodes probabilistic relationships among features of interest.

Theoretically, the supervised models perform better and provide more accurate detection rate when compared to unsupervised methods. The ability of encoding the interdependencies between features/ variables, combined with their ability to incorporate both prior knowledge and current data, supervised models return a better confidence score with the model output. However, in most practical sense, having labeled data is an unrealistic assumption.

- *Unsupervised Machine Learning for Anomaly Detection*

Unsupervised learning techniques do not require any manually labeled training data unlike in supervised. It presumes that most observations/ datapoints in a scenario present the normal behavior and only a small amount of percentage of observation is abnormal, thus anticipating that malicious observation is statistically different from the normal ones. Based on these above assumptions, frequently occurring groups of similar instances are classified as normal and infrequent data groups are categorized as malicious. The most popular unsupervised algorithms include GMMs, PCAs, K-means, Autoencoders, and hypothesis tests-based analysis [21].

Unsupervised anomaly detection algorithms are roughly categorized into the following main groups as illustrated in Figure 5. (1) Nearest-neighbor based techniques, (2) Clustering-based methods and (3) Statistical algorithms as shown in Figure 5. We shall be not going into the details of each of the category, since it is out of the scope of the thesis work. As mentioned earlier, we shall be using a machine learning based approach (association rule learning) and propose a new statistical-based approach to detect novel anomalies in parent-child process.

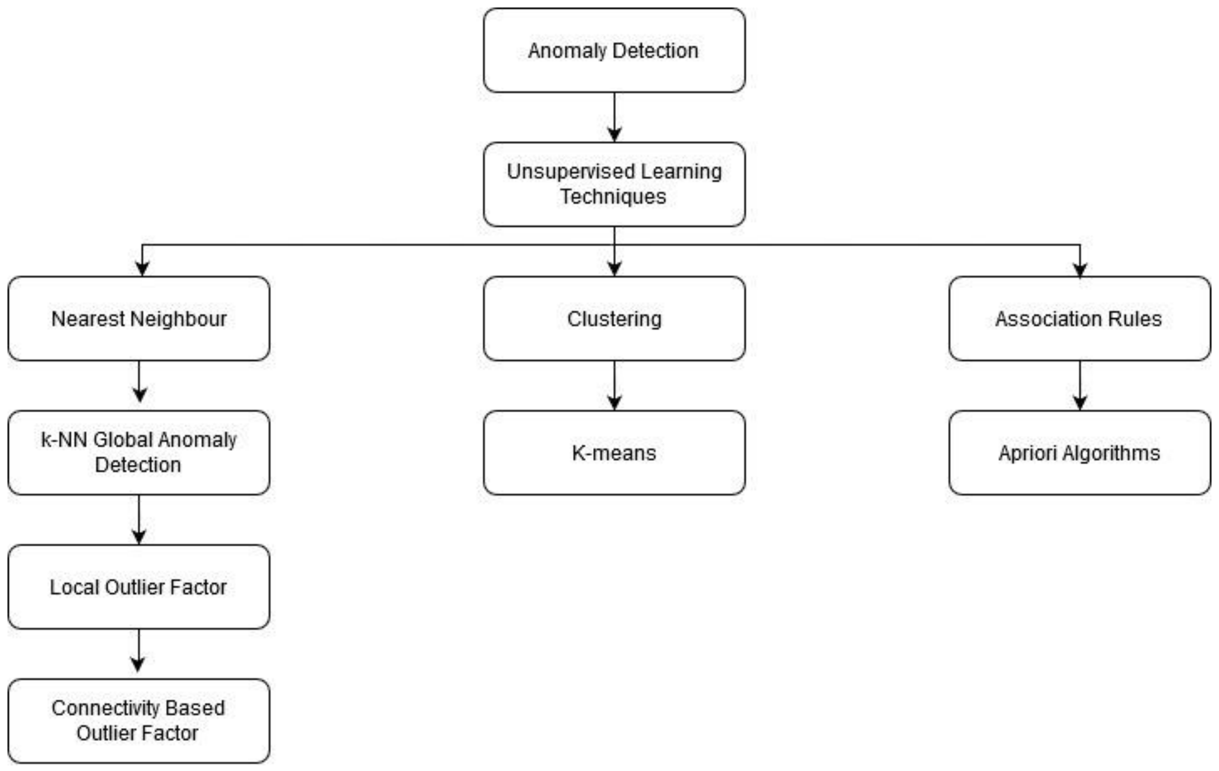


Figure 5: Categorization of Unsupervised machine learning for anomaly detection.

Chapter 3

Literature Survey and Related Work

In this chapter, we would review and present the literature review of the existing related work within the scope of this thesis work.

3.1 Earliest Works - IDS

One of the starting works on intrusion detection was a study by Jim Anderson [22] [11] which suggested ways on analyzing computer system audit data. The study introduced methods where data that is collected for performance analysis and other reasons was used for batch processing on audit data.

After Anderson's study, earlier works focused on developing procedures and algorithms for automating the offline security analysis of audit trails. The algorithms aim to provide ways for automated tools to help the security analysts perform assessments on batch of system activity (usually from last days). A similar project, conducted at SRI proposed using the audit data-logs, and suggested possible ways of creating automated security analysis of logs [23]. The work involved performing an extensive statistical analysis on audit data from the IBM systems running MVS (Multiple Virtual Storage) and VM (Virtual Machines). The objective of the study was to develop analytical and statistical techniques for screening computer system accounting data to detect user behavior indicative of intrusions.

IDES (Intrusion Detection Expert System), introduced by Lunt et al., 1992 [24] was a real time intrusion detection system that used both signatures and anomaly-based detection. The anomaly detection system of IDES worked by comparing the audit records against the normal profile of the user and that group the user was part of.

GrIDS (Graph based Intrusion Detection System), introduced by Chen et al., 1996 [25] used data source modules that run on each host and collects information to build a graph representation of an activity happening in the network, which is then compared with the normal profile to detect intrusions. GrIDS uses a central architecture and utilizes anomaly detection technique for intrusion detection.

Hyperview, introduced by Debar et al., 1992 [26] employs both - an artificial neural network component and an expert system component to detect intrusions. The expert system monitors the audit logs for known signatures of intrusions, while the neural network component learns the user behavior from the events and alerts for intrusions, when deviation is noted from the learned behavior.

RIPPER, by Lee et al., 1999 [27] uses data mining to generate automated and adaptive models for intrusion detection. The system extracts extensive feature set to describe the details of a network session and connection and applies association rule mining to learn the rules for normal and malicious behavior of the system. These rules are used for both signatures based and anomaly-based detection based on multiple algorithms used by RIPPER to generate patterns.

3.2 Rule Mining in Anomaly Detection

F. Silveira and Diot [28] proposed a tool called *URCA* that looks for anomalous flows by repeatedly eliminating subsets of normal flows and classifies the type of a detected anomaly. However, it can be computationally expensive as it requires continuous revaluation of detection system on different flow subsets.

Dowitcher proposed by S. Ranjan et al. [29] proposes a scalable system for worm detection and containment in backbone networks. The system automatically constructs a flow-filter from the intersection of suspicious attributes provided by different detectors leveraging suspicious attributes from an anomaly detector and study the anomaly extraction problem in more depth.

Dewaele et al. [30] creates numerous random projections of a traffic trace using sketches, then using Gamma laws, models the marginals of the sub traces and identify deviations in the parameters of the models as anomalies. The system tries to find possible anomalous source or destination IP addresses by taking the intersection of the addresses hashing into anomalous sub traces.

Lakhina et al. [31] propose a system to detect "network-wide anomalies" using SNMP data to highlight the origin-destination flow along which an anomaly existed. Li et al. [32], randomly aggregate flows as an alternative to origin-destination aggregation using sketches. It showed that more anomalies can be detected using random aggregation compared to the origin-destination aggregation in the PCA subspace anomaly detection method.

Lee and Stolfo in [33] propose a system for extracting interesting intrusion patterns from system calls and tcp dump logs using association rule learning. Vaarandi [34] introduces a tool called *LogHound* that provides an optimized implementation of Apriori and demonstrates how to summarize traffic flow records. Yoshida et al. [35] also identifies rare/ interesting events in traces from the MAWI traffic archive using frequent item-set mining. Chandola and Kumar [36]59describe rule mining-based approach to find a small set of highly frequent item-sets that can summarize a large set of flows.

Mahoney and Chan [37] identify rare events that could indicate anomalies in packet payload data using association rule mining (LERAD). In their approach they use 1999 DARPA/ Lincoln Laboratory traces to evaluate their approach and it aims at edge networks where mining rare events is possible. We can clearly see most of the work focused on network logs, however, we would be using similar approach from Mahoney and Chan [37] but on a host logs and relational data i.e. parent-child events and use the rule mining algorithm (LERAD) to find anomalous parent-child relations.

3.3 Dynamic Graph based Detection

In this section, we review previous approaches to detect anomalies in dynamic graphs and data streams. We referred Akoglu, Tong, and Koutra 2015 [38] and [39] for an extensive survey on graph-based anomaly detection. Since our proposed method focusses on data streams and dynamic graphs, we will cover techniques within that domain.

3.3.1 Anomaly Detection in Time Evolving Graphs

Anomaly detection in time evolving graphs is categorized based on the kind of anomalies, the following work tries to detect.

Anomaly detection in graph streams uses as input a series of graph snapshots over a period. Based on the types of anomaly following work is trying to detect, we categorize the related works as below:

Anomalous node detection: Sun, Tao, and Faloutsos 2006 [40] introduced a dynamic tensor analysis (DTA) method and by approximating the adjacency matrix of the current snapshot based on incremental matrix factorization, it spots nodes corresponding to rows with high reconstruction error. The system aims to create summary of high dimensional data and reveal hidden correlations.

Anomalous subgraph detection: Beutel et al. 2013 [41] proposed COPYCATCH that detects temporally bipartite cores that are ill-gotten "Likes" of Facebook for a given graph with timestamps on edges. The proposed method is aimed to catch lockstep Page Like patterns on Facebook by analyzing the social graph between users and pages and the times at which the edges in the graph (the Likes) were created. The system spots near bipartite cores where each node is connected to others in the same core density within a short time.

Anomalous event detection: Eswaran et al. 2018 [42] proposed a randomized sketching-based approach called SPOTLIGHT that provides high statistical guarantees that anomalous graphs are mapped at a distance from the normal graphs in the sketch space. The system, in close to real time, points to distinguish atypical graphs containing the sudden appearance or vanishing of huge dense subgraphs (e.g., close bicliques).

The system aims to detect anomalous graphs containing the sudden appearance or disappearance of large dense subgraphs (e.g., near bicliques) in near real-time.

Yoon et al. 2019 [43] proposed ANOMRANK, that uses a two-pronged approach defining two novel metrics for anomalousness tracking own version of a 'node score' (or node relevance) function which allows to detect sudden changes in node importance. The system detects suspicious changes in the structure of the graph, such as through the addition of edges between previously unrelated nodes in spam attacks and spots sudden changes in 1st and 2nd derivatives of PageRank.

Anomaly detection in edge streams: In edge streams, a stream of edges over time is used as input data for detection. We again categorize the related work based the type of anomaly they detect:

Anomalous node detection: Yu et al. 2013, in [44] proposed for a given edge stream, methods for dynamically determining anomalous hot spots in network stream. The work used principal component analysis (PCA) to maintain information about changes in neighborhoods of a graph and detects nodes whose egonets change suddenly and significantly.

Anomalous subgraph detection: Shin et al. 2017 [45] proposed *DenseStream and DenseAlert*, two incremental algorithms that maintains and updates dense subtensors, created within a short time, in a tensor stream (i.e., sequences of changes in a tensor), and spotting the sudden appearances of dense subtensors respectively

Anomalous edge detection: Ranshous et al. 2016 [46] focuses on sparsely connected parts of a graph and uses the global and local structural properties of a stream to build an outlier detection model. The paper uses Count-Min sketch for approximating these properties, and provide probabilistic error bounds on edge outlier scoring functions

Eswaran and Faloutsos 2018 [47] proposed a SEDANSPOT that identifies edge anomalies based on edge occurrence, preferential attachment, and mutual neighbors and scores the anomalousness of edges by considering the whole (sampled) graph, giving diminishing importance to far-off neighbors

Siddharth et al. 2019 [48] proposes a system to detect suddenly arriving groups of suspiciously similar edges or micro-cluster anomalies, such as lockstep behavior along with detecting DDoS (denial of service) attacks in network traffic data. The paper presents two algorithms MIDAS, and MIDAS-R, with one having temporal decay factor for time relevance in data.

Our proposed work is more related to the anomaly detection problem in edge stream, where we treat the problem of parent-child anomaly as a relationship and learn the behavior using statistical properties of the relationship in graph to detect anomalous edges.

Chapter 4

Methodology and Proposed Work

In this chapter we shall be discussing LERAD (Learning rules for Anomaly detection) algorithm and presenting our novel algorithm for detecting parent-child anomalies, since data in events logs cannot always be treated as points lying in a multi-dimensional space independently, rather they exhibit inter-dependencies which should be accounted for during the anomaly detection process. Based on this idea we would discussing on the two algorithms.

Approach 1: Rule Learning based Approach

4.1 Data Mining and Association Rule Mining

Data mining is the process where data is analyzed from different perspectives and then summarized into useful information [49]. Data mining allows analyzing data, categorizing it, and summarizing the relationships among data attributes. Technically, data mining can be described as the process of finding patterns and correlations within large relational databases.

Primarily used to discover interesting relationships, association rules are basically if/ then statements that are used to discover/ establish relations between unrelated data in a database, relational database, or other information repository i.e. to find the hidden relationships between the elements that occur frequently together. Use cases of association rules are manifold including classification, marketing, basket data analysis, recommender systems, clustering, and loss-leader analysis etc. In 1993, article from Agrawal et al. [50] popularized the concept of association rules due to the which it is widely used within the Data Mining field and is regularly cited in academic community. However, what is now called "association rules" was introduced already in the 1966 paper on GUHA, a general data mining method developed by Petr Hájek et al [51].

The paper by Aggarwal et al. [50] defines the association rule problem as:

Let $\boldsymbol{I} = \{\boldsymbol{i_1}, \boldsymbol{i_2}, \dots\dots, \boldsymbol{i_n}\}$ be a set of $\boldsymbol{n}$ binary attributes called items.

Let $\boldsymbol{D} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \dots\dots, \boldsymbol{t_n}\}$ be a set of transactions called the database.

Each transaction in $\boldsymbol{D}$ has unique transaction ID and contains a subset of the items in $\boldsymbol{I}$.

A rule is usually is defined in the below form:

$\boldsymbol{X} => \boldsymbol{Y}$, where $\boldsymbol{X}, \boldsymbol{Y} \subseteq \boldsymbol{I}$

In the paper, a rule is defined only as relation between a set and a single itemset,

$\boldsymbol{X} => \boldsymbol{i_j}, \boldsymbol{for}\ \boldsymbol{i_j} \in \boldsymbol{I}$

Every rule is composed by two different sets of items, also known as item, sets $\boldsymbol{X\ and\ Y}$ and, where $\boldsymbol{X}$ is called the antecedent and $\boldsymbol{Y}$ the consequent.

In rule mining there are basically three major quantitative measurements that must be taken into consideration, the support, the confidence and lift. These parameters are used widely to decide on the importance or quality of a rule generated.

Support is an indication of how regularly the items appear in the dataset. Mathematically, support is defined as the fraction of the transactions having item X and Y with total number of transactions in which the item set occurs.

$$\text{Support } \{x\} \rightarrow \{y\} = \frac{\boldsymbol{Transactions\ containing\ both\ x\ and\ y}}{\boldsymbol{Total\ number\ of\ Transactions}}$$

Confidence defined the number of times the if-then statements of associations rules are found true. Confidence can hence be defined as the conditional probability of occurrence of consequent given the antecedent.

$$\text{Confidence } \{x\} \rightarrow \{y\} = \frac{\boldsymbol{Transactions\ containing\ both\ x\ and\ y}}{\boldsymbol{Transactions\ containing\ x}}$$

Lift is used to compare the actual confidence with expected confidence. This can be described given item X was purchased, how likely item Y is purchased, at the same time controlling the popularity of item Y. So, mathematically,

$$\text{Lift}\{x\} \rightarrow \{y\} = \frac{\frac{(\boldsymbol{Transactions\ containing\ both\ x\ and\ y})}{(\boldsymbol{Transactions\ containing\ x})}}{(\boldsymbol{Fracation\ of\ Transactions\ containing\ y})}$$

4.2 Learning rules for Anomaly detection algorithm (LERAD)

LERAD generates rules from arbitrary combinations of nominal attributes, eliminating the need to select rules in an ad-hoc fashion. The goal of LERAD is to find conditional rules that can identify unexpected/ rare events in a time-series of tuples of nominal (unordered) attributes (e.g. packet field values, or words in a TCP session), as introduced in the original paper by Mahoney and Chan [37]. Given two tuples, the time series exhibits long range dependencies; the number of matching attribute values decreases as the time interval between the tuples increases.

Rules generated have the general form:

"if $\boldsymbol{A_1}$ = $\boldsymbol{v_1}$ and $\boldsymbol{A_2}$ = $\boldsymbol{v_2}$ and ... and $\boldsymbol{A_k}$ = $\boldsymbol{v_k}$ then $\boldsymbol{A_{k+1}}$ € $\boldsymbol{V}$ = {$\boldsymbol{v_{k+1}}$, $\boldsymbol{v_{k+2}}$, …, $\boldsymbol{v_{k+r}}$}$^{\boldsymbol{n}}$, where the $\boldsymbol{A's}$ are attributes and $\boldsymbol{v}'$s are values.

From this huge rule space that is created, LERAD aims to select the rules that would generate the highest anomaly scores, i.e. those that have high n and low r, where n is the number of training instances that satisfy the antecedent ($\boldsymbol{A_1}$ = $\boldsymbol{v_1}$, and $\boldsymbol{A_k}$ = $\boldsymbol{v_k}$, $\boldsymbol{k\ r = |V|}$), and r is the number of allowed values in the consequent. This goal is different from earlier rule mining algorithms such as [52] or [53], which have the goal of finding rules that predict a single value in the consequent with high probability (i.e. high confidence).

Though the target for rule mining algorithms, specifically the above three, is to find rules having support value (large n), LERAD differs a bit, since regardless of the distribution within V, it tries to find rules with small r.

For example, one rule might be "if port = 23 then name € {"HELO", "EHLO"}". In training phase, LERAD checks the number of observations where port = 23, counts n, and records V i.e. the set of values for name. While in testing phase, when LERAD observes an instance with port = 23 but the corresponding name value is not "HELO" or "EHLO", then an anomaly score of *tn/r* is assigned, where $\boldsymbol{r = |V| = 2}$, and *t* is the time since the rule was last violated. This process is repeated for other rules as well and the total score assigned to a instance during testing would the summation of the anomaly scores assigned by each rule (in the rule set) for which the antecedent is matched but the consequent is not.

4.2.1 Rule Learning

LERAD aims to generate good rules (high n/r, aims to keep r small) but challenge is searching the huge rule space, which grows exponentially with the increasing number of attributes. LERAD uses a randomized algorithm [37] to generate candidate rules, then tests them on increasingly large subsets of the training data, discarding the redundant rules and those that do not satisfy the constraint of high n/r. The steps are as follows:

1. haphazardly generate rules with n/r = 2/1 on pairs of training instances.
2. discard redundant rules in favor of those with those with higher n/r on a larger training sample, *S*.
3. get rid of the rules that show poor performance on the complete training set (specially, where r increases towards the end). LERAD makes two iterations over the training data, first to sample the training instances as in step 1 and 2, and then finally to train the rules as per step 3.

4.2.2 Candidate rule generation

The candidate rules are made by randomly selecting pairs of instances from the training set and finding rules that satisfy both instances (one or more attributes have the same value in both instances). In such case, one attribute becomes the consequent and any subset of the remaining attributes can become the antecedent. For example, we have two training examples as given below:

port = 80 *word1 = word.exe* *word2 = /* *word3 = Microsoft*

port = 80 *word1 = word.exe* *word2 = /wmic.exe* *word3 = Microsoft*

There are three matching attributes in the example above namely port, word1, and word3. Some of the 12 possible rules with *n/r = 2/1* are:

- *word1 = "word.exe"*
- *if word3 = "Microsoft" then port = 80*
- *if port = 80 and word1 = "word.exe" then word3 = "Microsoft"*

In general, if there are $\boldsymbol{k}$ matching attributes, then there are $\boldsymbol{k}$ possible consequents and $\boldsymbol{2^{k-1}}$ possible subsets of the remaining attributes to form the antecedent, allowing for

$\boldsymbol{k2^{k-1}}$possible rules. LERAD does this by randomly picking a subset of these rules as candidates. LERAD does this as detailed in the following algorithm [37]:

- randomly select a sample S training instances
- repeat L times
- haphazardly select a pair of training instances from S
- randomly order the $\boldsymbol{k}$ matching attributes in a sequence, but not more than $\boldsymbol{k_{max}}$
- generate $\boldsymbol{k}$ rules using 1 through $\boldsymbol{k}$ attributes, making the first one the consequent and the others the antecedent.

In the original paper by Mahoney and Chan, it makes little difference whether the random pairs are selected from S or from the full training set. In our experiment we used |S| = 100000, and notices it makes little difference on increasing the |S|.

4.2.3 Removing duplicate rules

The second major step in LERAD is to get rid of rules generated that really do not provide any new information about a small sample training set, S. When two rules predict the same values, LERAD will keep the one with the higher n/r (when its trained-on S), or in case of a tie in scores, the one with fewer conditions in the antecedent. The algorithm for deduping is as follows:

- Sort the candidate rules by descending n/r on S, or by ascending size of the antecedent in case of ties.
- For each rule
    - For each sample in S
        - if the sample instance satisfies the antecedent, then mark the consequent value in S unless already marked
        - remove the rule if no new values in S could be marked.

Approach 2: Dynamic Graph based Approach

In this section, we would be presenting a novel way for detecting parent-child anomalies. We model the problem of detecting parent child anomalies in streaming data as relational models that are built over time. Our approach considers the dynamic behavior of changing relationships in attributes of a graph. We use a count min sketch-based approach to build relationship and score the streaming data on logarithmic and probabilistic scales.

4.3 Graphs and why graphs?

A graph is an abstract data type that requires two basic building blocks: nodes and vertices as shown in Figure 6. A graph utilizes the basic idea of using vertices to establish relationships between pairs of nodes. Many real-world relationships specially with interdependencies or hierarchical structure are best modelled using graph structures.

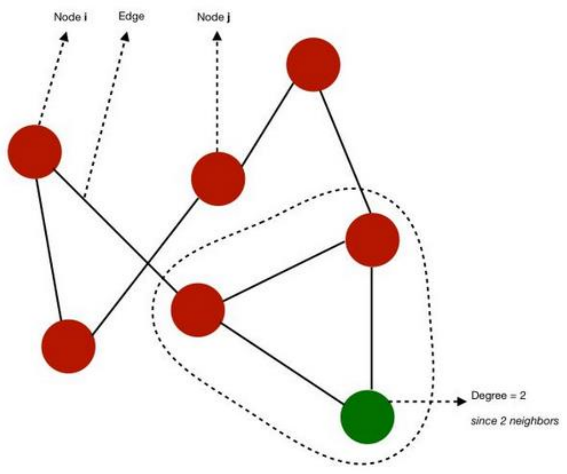


Figure 6: Graph shows nodes and vertices

A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ has the following attributes and is composed of:

- nodes, also called vertices, $\mathbf{V} = \mathbf{1}, \ldots, \mathbf{n}$
- edges $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$
- An edge $(\boldsymbol{i}, \boldsymbol{j}) \in \boldsymbol{E}$ links nodes i and j, where i and j would be called as neighbors.
- number of neighbors of a node is presented by its *degree*

Dynamic graphs allow modification to the structure and/or attributes of the graph over a period, and thus is an efficient way to model an evolving set of entities and their changing interactions/ relationships. A common example can be social media platforms like Facebook/ Twitter, where vertices of the graph are the users and the edges can represent friendship or followers, posts, messages, or a bunch of other phenomena.

We also highlight below the importance of using a graph-based approach for our use case of detecting anomalous parent-child behavior:

- Inter-dependent nature of the data or the problem statement: Data objects or system logs in our case are often related to each other and exhibit multiple dependencies. In fact, most relational data can be thought of as inter-dependent, which necessitates to account for related objects in finding anomalies, especially since we are interested find parent-child based anomalies.
- Powerful representation: Graphs are intuitively better in representing the inter-dependencies by the introduction of links (or edges) between the related entities and thus captures the relationship effectively.
- Robustness: Graphs are much more immune to adversarial attacks, since an attacker can be assumed to possess limited knowledge of the infrastructure and even if he/ she can modify a transaction, it’s much harder to fit in without complete knowledge of all characteristics and interdependencies of the operations.

4.4 Streaming Data Structures

In general, dynamic graph can be seen as made up of a sequence or streams of static graphs, and analysis is done by comparing neighboring graphs in the stream, however, this approach has many limitations for outlier detection. Firstly, it requires storing the entire graph in memory for offline analysis, which is unfeasible with current scale of nodes and edges that a graph can grow to in real world data. Secondly, most of the past or historical information is lost in streams as the most recent graphs in the stream are considered for detection purposes. Thirdly, comparing graphs at global level misses the element of looking into changing nature of attributes in the graph.

Since edge stream mining provides a more fine-grained analysis of the changes, we use that approach to model the relations within the graph. However, addressing memory constraints and historical information loss issues is still a challenging problem. To address this, we use Count-Min sketch [54] to approximate the structural properties of the stream that are relevant to building the model. Using this approximation, we can model probabilistic relationships of the nodes.

In our approach we use Count-Min sketch [54] to statistically score outliers/ anomalous events. A record or edge (representing a relation) is marked as anomalous based on the likelihood of its occurrence when its score exceeds a user-defined threshold. Simply put, an anomaly is realized as a data point or a group of objects that is rare (for example, atypical combination of categorical attribute values), and/ or surprising (for example, data points that don't align with our statistical model).

4.5 Count-Min Sketch (CMS)

The Count-Min (CM) Sketch is a compact summary data structure capable of representing a high-dimensional vector and answering queries on this vector, point queries and dot product queries, with strong accuracy guarantees [54] and space efficient using probabilistic techniques [54] [55]. Since the data structure can easily process updates in the form of additions or subtractions to dimensions of the vector (which may correspond to insertions, deletions, queries (or other transactions), it is capable of working over streams of updates, at extremely high rates. The Count-Min sketch (CMS) was first proposed in 2005 [54] with goal being to provide a simple sketch data structure with a precise characterization of the dependence on the input parameters. The simplicity of creating the sketch and easy probing for queries has led to its wide use in disparate areas since its initial proposal.

In an ideal scenario for retrieving frequency of any streaming data we would use hash table as we can store the hash values in the Hash table and retrieve them easily at $\boldsymbol{O(1)}$. However, this approach is highly inefficient because in the above approach we end-up storing all the data in the hash tables which means there is liner memory usage for very large (infinite) streaming of data.

To tackle this very memory inefficiency, we use count min sketch to calculate the frequency in sub-liner space , because in this case we don't store the complete values of data stream , instead the CMS approach uses a matrix to compute the frequency, where number of rows are the number of Hash functions that are used and columns are number of outcome of the hash functions.

The CM sketch is simply an array of counters of width $\boldsymbol{w}$ and depth $\boldsymbol{d}$, $\mathbf{CM}[\mathbf{1},\mathbf{1}]\ldots\mathbf{CM}[\mathbf{d},\mathbf{w}]$. Each entry of the array is initially zero. Additionally, $\boldsymbol{d}$ hash functions $\boldsymbol{h1}\ldots\boldsymbol{hd} : \{\mathbf{1}\ldots\boldsymbol{n}\} \rightarrow \{\mathbf{1}\ldots\boldsymbol{w}\}$ are chosen uniformly at random from a pairwise-independent family. Once $\boldsymbol{w}$ and $\boldsymbol{d}$ are chosen, the memory space required is fixed, ensuring constant space guarantees: the CMS data structure is represented by $\boldsymbol{wd}$ counters and $\boldsymbol{d}$ hash functions (which can each be represented in O(1) machine words [56].

4.6 Data Model

Given a continuous edge stream $\boldsymbol{E} = < (\boldsymbol{u}, \boldsymbol{v}), \ldots\ldots >$, where $\boldsymbol{u}$ ***and*** $\boldsymbol{v}$ are the vertices the edge is incident upon. Every edge in the stream is directed, un-weighted and represents the insertion of a new single edge into the graph. The vertices in the graph are the entities or nodes among which we want to establish a relationship, for example, vertices can be filename and parent filename etc. Edges are labeled as the concatenation of the two vertices they are incident upon and basically indicate the relationship, thus the label of $\boldsymbol{e} = (\boldsymbol{u}, \boldsymbol{v})\ is\ \boldsymbol{l}(\boldsymbol{e}) = \boldsymbol{l}(\boldsymbol{u}) \oplus \boldsymbol{l}(\boldsymbol{v})$, where $\oplus$ is the concatenation operator, and represents the relationship between $\boldsymbol{u}$ ***and*** $\boldsymbol{v}$. We try to assign weights to the edges in a way that it is indicative of their expected anomalousness.

4.6.1 Edge Scoring

Probability based edge scoring

Conditional probability is defined as a measure of the probability of an event occurring with relationship to one or more events (given that this another event has occurred) [57]60If suppose the event we are interested in is X and the event Y is assumed to have already occurred, "the conditional probability of X given Y" is P (X | Y).

$$\boldsymbol{P}(\boldsymbol{X} \mid \boldsymbol{Y}) = \frac{\boldsymbol{P}(\boldsymbol{X} \cap \boldsymbol{Y})}{\boldsymbol{P}(\boldsymbol{Y})}$$

, where $\boldsymbol{P}(\boldsymbol{X} \cap \boldsymbol{Y})$ is the probability that both events X and Y occur. In our case we calculate the conditional probability distribution.

If the conditional distribution of X, given Y and Y is a continuous distribution, then its probability density function is known as the conditional density function. The conditional probability, for discrete random variables, function of X, given Y = y can be written according to its definition as [58]:

$$\boldsymbol{P(X = x \mid Y = y) = \frac{P(\{X = x\} \cap P\{Y = y\})}{P(Y = y)}}$$

We build relational model based on conditional probability for a combination of attributes, where we map the relation between node $\boldsymbol{u}\ and\ \boldsymbol{v}$ as $\boldsymbol{l(e) = l(u) \oplus l(v)}$, where e is the edge in data stream. The score for each edge is given by $\boldsymbol{P_n(u, v)}$:

$$\boldsymbol{P_n(u, v) = \frac{\sum_{i=0}^{n} \ P(\{X = u_i\} \cap P\{Y = v_i\})}{\sum_{i=0}^{n} \ P(Y = v_i)}}$$

, where the $\boldsymbol{P_n(u, v)}$ is the score of the $\boldsymbol{n^{th}}$edge stream. So $\boldsymbol{P_n(u, v)}$ indicates how likely is the edge $\boldsymbol{e = (u, v)}$ to occur given node $\boldsymbol{v}$ occurs.

Logarithmic edge scoring

A logarithmic scale (or log scale) represents in a concise way a numerical data over a wide range of values, which is achieved by displaying the number on a non-linear scale. Similar to probability-based edge scoring, for an edge $\boldsymbol{e = (u, v)}$, the log score for a relation $\boldsymbol{l(e) = l(u) \oplus l(v)}$, is defined by $\boldsymbol{L_{uv}}$ and given by:

$$\boldsymbol{L_n(u, v) = \ln \left(\sum_{i=0}^{n} l(e_i) \right)}$$

, where the $\boldsymbol{L_n(u, v)}$ is the score of the $\boldsymbol{n^{th}}$edge stream and $\boldsymbol{l(e) = l(u) \oplus l(v)}$, . So $\boldsymbol{L_n(u, v)}$ transforms the occurrence of an edge into non-linear logarithmic space.

Further, we wanted to extract a numeric value for the relevance of each relation built, in an unsupervised setting and for that we use Laplacian score [59] that is designed to evaluate the importance of every feature which is achieved by optimizing cluster coherence by spectral gap of the corresponding affinity matrix. The calculated Laplacian score is used to create the penalizing weight that would be described later.

Laplacian score is based on the observation that data from the same class is often close to each other and thus it is possible to evaluate the importance of a given feature by its power of locality preserving. The method consists in embedding the data on a nearest neighbor graph by using an arbitrary distance measure and then calculating a weight matrix [60]. A Laplacian score is then calculated for every feature (relations built, in our case namely $\boldsymbol{P_n(u,v)}$ and $\boldsymbol{L_n(u,v)}$) in the data and will have the property that smallest values correspond to the most important dimensions. However, to select a subset of features another clustering algorithm (e.g. k-means) is usually applied a-posteriori to select the best performing group. The Algorithm used is presented in [59] and is given below (the complete details of the algorithm can be referred from the paper [59]

4.6.2 Laplacian Feature Scoring

Let $\boldsymbol{L_r}$ denote the Laplacian Score of the r-th feature. Let $\boldsymbol{f_{ri}}$ denote the i-th sample of the r-th feature, $\boldsymbol{i = 1, \dots\ , m}$. The Laplacian Score algorithm is given in [59]:

1. Construct a nearest neighbor graph G with m nodes. The i-th node corresponds to $\boldsymbol{x_i}$. We put an edge between nodes **i** and **j** if $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are ”close”, i.e. $\boldsymbol{x_i}$ is among **k** nearest neighbors of $\boldsymbol{x_i}$ or $\boldsymbol{x_j}$ is among k nearest neighbors of $\boldsymbol{x_i}$. When the label information is available, one can put an edge between two nodes sharing the same label.
2. If nodes **i** and **j** are connected, put $\boldsymbol{S_{ij}}$ = $\boldsymbol{e}^{\frac{|xi-xj|^2}{t}}$, where t is a suitable constant. Otherwise, put $\boldsymbol{S_{ij}}$ = 0. The weight matrix **S** of the graph models the local structure of the data space.
3. For the r-th feature, we define:
   $f_r = [f_{r1}, f_{r2}, \dots, f_{rm}]^T, D = diag(S1), 1 = [1, \dots, 1]^T, L = D - S,$
   where the matrix L is also called the graph Laplacian. Let
   $$f_r^{\sim} = f_r - \frac{f_r^T\, D1}{1^T\, D1}\, 1$$
4. Compute the Laplacian Score of the r-th feature as follows:
   $$L_r = \frac{f_r^{\sim T}\, L f_r^{\sim}}{f_r^{\sim T}\, D f_r^{\sim}}$$

Using the Laplacian score for each feature or relation, we define a penalizing weight of a relation, which is given by P_r,

$$P_r = \frac{L_r}{\min L_{rn}}$$

, where $L_{rn} = [L_{r1,}\, L_{r2}, \dots, L_{rn}]$ i.e. set of Laplacian scores of features/ relations.

For each edge stream score $L_n(u,v)$ *and* $P_n(u,v)$, we penalize the score by dividing with the P_r, for each relation where $L_n(u,v) < \mathit{standard\ deviation}\ \big(L\ (u,v)\big)\ \mathit{and}\ P_n(u,v) < \mathit{standard\ deviation}\ \big(P\ (u,v)\big)$ and obtain the corrected score $L_{n'}(u,v)$ *and* $P_{n'}(u,v)$ for each edge stream.

Final anomaly score F_n, is given by:

$$F_n(u,v) = \frac{\sum_{i=0}^{n}\ (L_{n'}(u,v) + P_{n'}(u,v))}{\mathrm{n}}$$

, where n is the number of relations used.

4.6.3 Community detection and Scoring

Now that for each edge stream we have a score, $F_n(u,v)$, one major consideration is to look at the noise that is generated and control the false positives (FPs) that we have in the results. The number of FPs is a major issue in any unsupervised learning detection. For, our threshold parameter (details in Chapter 6), we observe that there are ~17k events that are flagged as anomalies, where most of them are one-off system activities that obtained low score.

Hence to reduce the number of events a security analyst needs to look at, we create communities for the events in the dataset. The communities in our case is defined as a process tree, where the parent process spawns another process and so on as showing in Figure 7. We create communities based on the process tree and calculate a score for the community, based on the score calculated in earlier section (assuming that in a malicious process, each process would be breaking the multiple relations threshold) and define threshold for describing malicious communities. The score is calculated as the total relations broken $L_n(u,v) < \mathit{standard\ deviation}\ \big(L\ (u,v)\big)\ \mathit{and}\ P_n(u,v) < \mathit{standard\ deviation}\ \big(P\ (u,v)\big)$ within a community, divided by the depth of the process tree.

This approach reduced the noise by ~200x for an analyst investigating the detections.

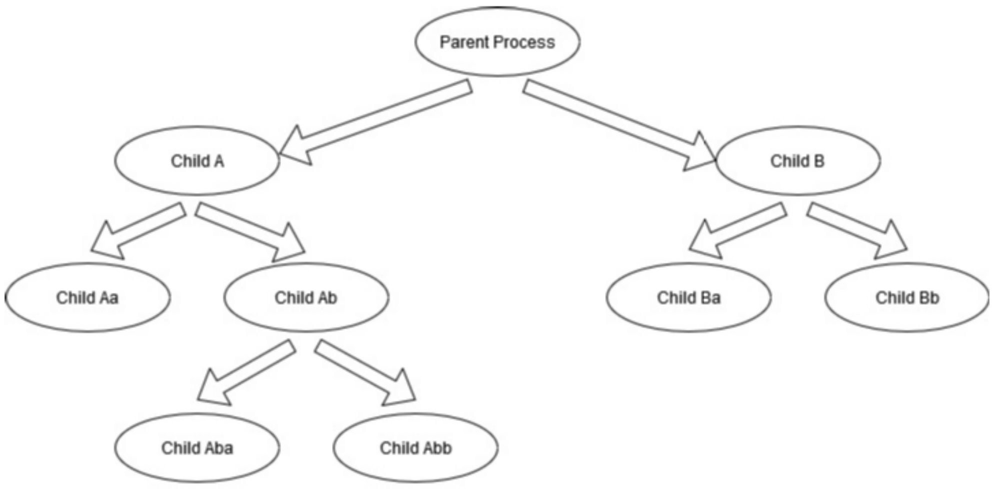


Figure 7: Figure showing process tree to define communities and score communities.

Chapter 5

System Setup and Methodology

In this Chapter, we would present more details about the engineering and system setup for the project, the dataset preparation and the general pipeline of the service created for anomaly detection. We will be covering the necessary and important details of how the service is setup and how the features for the experiments are being fetched. In our case the dataset prepared would be specially targeted to hunt for malicious parent-child anomalies and by doing so, we try to detect "living of the land" attacks that is earlier described in Chapter 2.

5.1 General Setup and Data Source

In this section, we elaborate on the general setup of the entire service and how the system has been setup. Since most of the components where event logs are captured, processed is an intellectual property (IP) of ReaQta and hence cannot be elaborated, we would try and describe the setup from a macroscopic view in this work.

The general setup as also represented in the Figure 7, the entire system can be seen as built of 4 major components: Driver/ Kernel setup, Agent, Event Hive, Detection service.

Driver/ kernel setup:

This is primarily a kernel level implementation of callbacks to generate the data source for ReaQta Hive, the core part of ReaQta service including the Nano OS (live hypervisor-based monitoring of the OS from outside). We shall, however, be not covering details about ReaQta Hive, since we use the service only to gather the data for our analysis though we shall look briefly on the driver routines that fetch the data for Event Hive service. Driver support routines are routines that the Windows operating system provides for kernel-mode drivers to use. Drivers do not use Microsoft Win32 routines; instead, they use the driver support routines that are organized by kernel-mode managers and libraries.

For instance, the "PsSetCreateProcessNotifyRoutineEx" 60[61] routine registers or removes a callback routine which simply notifies the caller when a process is created or exits. Highest-level drivers can call "PsSetCreateProcessNotifyRoutineEx" to register a PCREATE_PROCESS_NOTIFY_ROUTINE_EX routine. An installable file system (IFS) or highest-level system-profiling driver may register a process-creation callback routine to track which processes are created or deleted against the driver's internal state across the system.

Agent:

This is small service running of target endpoints that collects the data from callbacks and other sources. In case, any data is missing in the collection phase the Agent generates the information from other sources and prepares the data to be sent to ReaQta Event Hive service.

Event Hive:

This has primarily a Cassandra database, an open source NoSQL database management system, that stores the event information fetched from the Agent. Details of the implementation of this is an IP of ReaQta and outside the scope of the work presented in the thesis.

Detection Service:

This is the main part where our anomaly detection engine primarily resides. This is a python based microservice build on top of Flask framework[1] , that fetches the data from Event Hive.

The detection system basically has two components: one for transformation and other for detection. The transformer part ingests the feed from Event Hive and prepares the events with attributes that are required for training and transforming event with parent child relations, that is described in more details in the next section. The detection part is where our algorithm for rule generation and statistical detection sits and consumes the data post transformation.

1. https://flask.palletsprojects.com/en/1.1.x/

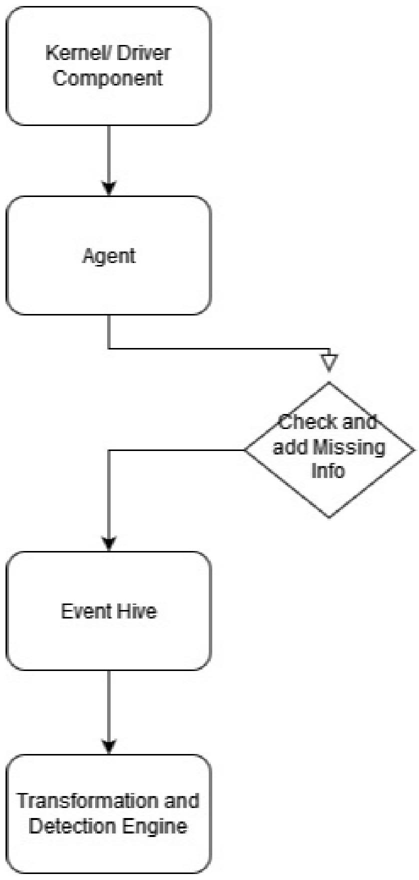

Kernel/ Driver Component
Agent
Check and add Missing Info
Event Hive
Transformation and Detection Engine

Figure 7: Diagram showing system flow for Detection System

5.2 Data Engineering and Preprocessing

Data from Event-Hive before being fed to the detection system is transformed into parent-child events. While the payload from Event Hive has many attributes associated with an event, we necessarily pull the attributes described in Table 1. The parent child relationship is built on an infrastructure level using fields “lerad_id” and “lerad_parent_id”, which we build using 3 data sources, namely, “endpoint_id”, “pid” and “start_time”. The details of the fields are covered in Table 1. The combination ensures that parent child events can be identified on an infrastructure level, since the rules and model is built for the entire infrastructure level.

Once the event transformation is completed, the parent-child relationships are stored in PostgreSQL Database.

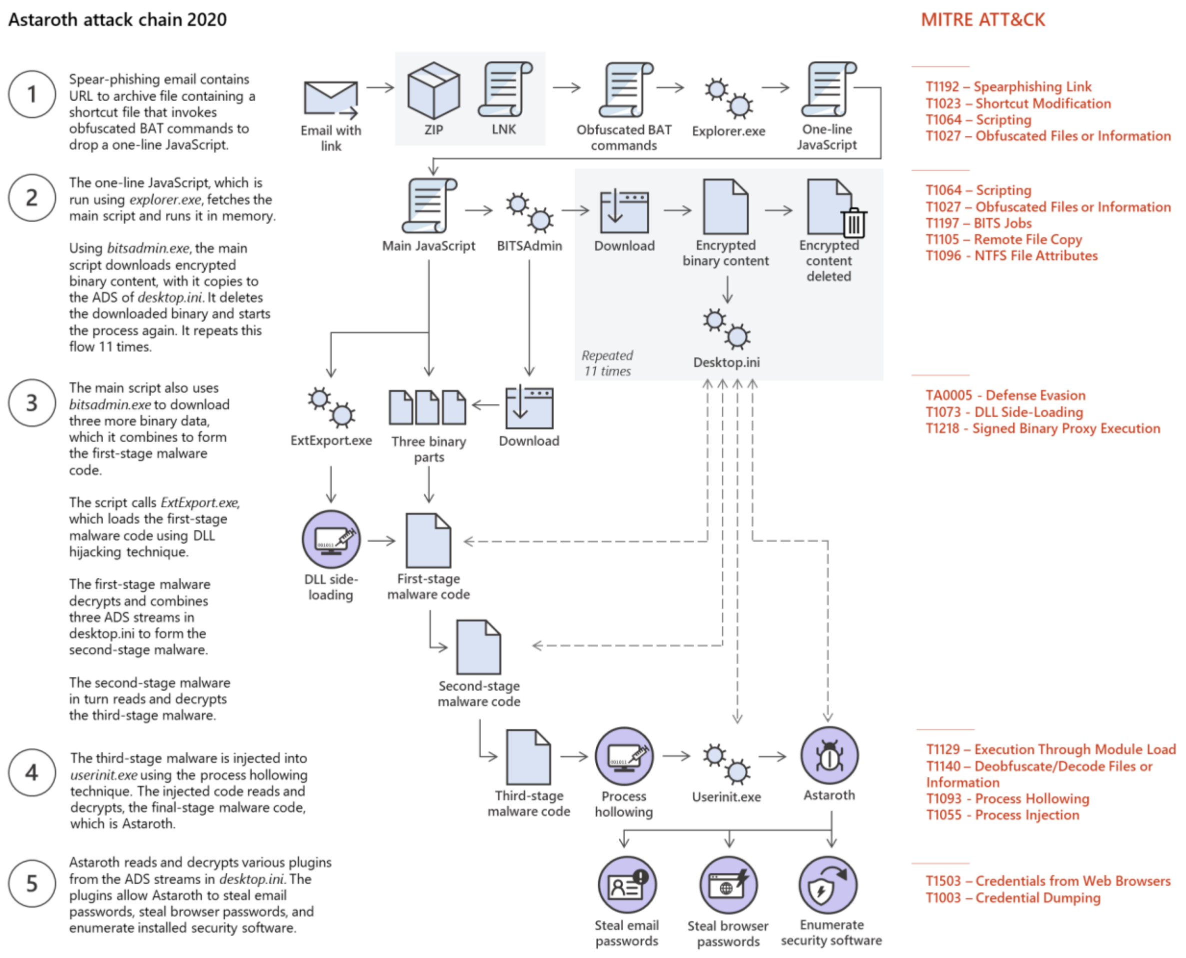
lerad_id	Unique ID and is a combination of 3 attributes to build child identity. Includes: endpoint_id, pid, and start_time
lerad_parent_id	Unique ID and is a combination of 3 attributes to build parent identity. Includes: endpoint_id, ppid, and pstart_time
pstart_time	Unix timestamp of parent process start time
start_time	Unix timestamp of process start time
cmd_line	Command line argument associated with the process
pid	Process ID of the event
ppid	Parent process id of the event
hive_event_id	Unique ID from Event Hive
hive_endpoint_id	Endpoint ID
hive_event_local_id	Local event ID
path_abs	Encodes value from path of the directory where the process started from.
privilege_level	Privilege level of execution (possible values – Low, Medium, High)
user_sid	User serial ID
path	Path of directory of execution
size	Size (in bytes) if the file
md5	MD5 hash of the file
sha1	SHA1 hash of the file
sha256	SHA2 hash of the file
arch	Program architecture (x32 or x64)0
filename	Filename of the executable
cert_issuer	Certificate issuer
cert_signer	Certificate signer
cert_expired	Boolean value for if the certificate is expired or not
cert_trusted	Boolean value for if the certificate is trusted or not

Table 1: Fields for parent-child transformation and training

5.3 Dataset Generation

The problem of having opensource and labelled dataset was evident in our case, since there is no labelled dataset that could be used to build parent-child relationships. We evaluated the popular publicly available datasets such as DARPA [62], KDD [63], NSL-KDD and ADFA-LD [64] as they are widely used as benchmarks, however, most of these are not HIDS logs, besides being unusable to get parent-child relationships from them. Further, we evaluated the dataset [65], which is built , however, the dataset is too small and inconsistent to be used. Therefore, we decided to extract data from one of the customers to build and train the model and execute three malwares know to utilize "Lotl" tactics and evaluate the detection capability based on that (the evaluations is aimed to prove the hypothesis and not set/ compare against benchmark).

We executed Guildma (also known as Astaroth) malware and extracted logs created by the malware's execution. Written in Delphi, this malware (targeted towards Latin banks), uses some novel innovative execution and attack techniques. In Guildma, the attack is orchestrated by its C&C server rather than storing the fake pop-up windows it uses within the binary. One of the defining characteristics of Guildma's distribution chains is using tools already present on the system, often in new and unusual ways aka Lotl techniques. The detailed breakup of the TTPs used by Astaroth as reported by Microsoft security team is also shown below [66].


Astaroth attack chain 2020

1 Spear-phishing email contains URL to archive file containing a shortcut file that invokes obfuscated BAT commands to drop a one-line JavaScript.

2 The one-line JavaScript, which is run using *explorer.exe*, fetches the main script and runs it in memory.

Using *bitsadmin.exe*, the main script downloads encrypted binary content, with it copies to the ADS of *desktop.ini*. It deletes the downloaded binary and starts the process again. It repeats this flow 11 times.

3 The main script also uses *bitsadmin.exe* to download three more binary data, which it combines to form the first-stage malware code.

The script calls *ExtExport.exe*, which loads the first-stage malware code using DLL hijacking technique.

The first-stage malware decrypts and combines three ADS streams in desktop.ini to form the second-stage malware.

The second-stage malware in turn reads and decrypts the third-stage malware.

4 The third-stage malware is injected into *userinit.exe* using the process hollowing technique. The injected code reads and decrypts, the final-stage malware code, which is Astaroth.

5 Astaroth reads and decrypts various plugins from the ADS streams in *desktop.ini*. The plugins allow Astaroth to steal email passwords, steal browser passwords, and enumerate installed security software.

MITRE ATT&CK

T1192 – Spearphishing Link
T1023 – Shortcut Modification
T1064 – Scripting
T1027 – Obfuscated Files or Information

T1064 – Scripting
T1027 – Obfuscated Files or Information
T1197 – BITS Jobs
T1105 – Remote File Copy
T1096 - NTFS File Attributes

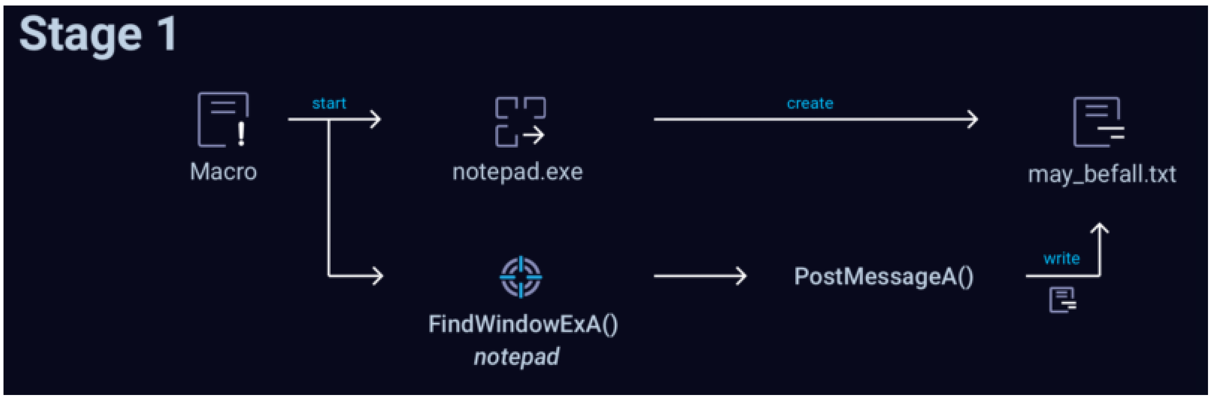
TA0005 - Defense Evasion
T1073 - DLL Side-Loading
T1218 - Signed Binary Proxy Execution

T1129 – Execution Through Module Load
T1140 – Deobfuscate/Decode Files or Information
T1093 - Process Hollowing
T1055 - Process Injection

T1503 – Credentials from Web Browsers
T1003 – Credential Dumping

Figure 8: Astaroth attack chain 2020

We picked 2 more malwares that were real incidents namely Dridex malware [67] and Azorult malware [68], both affecting the banking industry and known to use “Lotl” techniques.



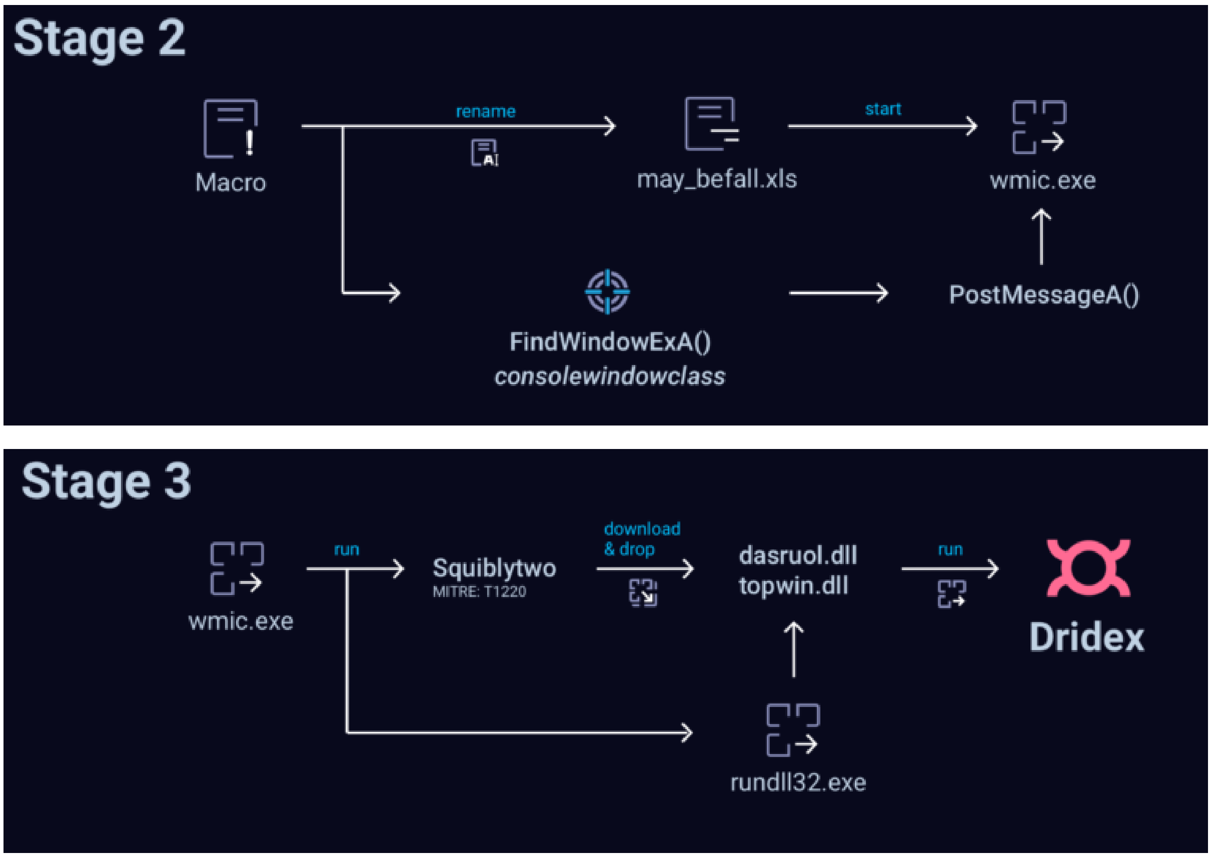


Figure 9: Stages of execution of Dridex

Chapter 6

Evaluation and Outcome

We performed our analysis on a customer data and used actual malware (known to use Lotl techniques) execution logs to test the detection capabilities of the proposed system. The dataset consists of 1788561 parent-child transformed events and was used in training, building the model. The logs were extracted for 44 days.

6.1 Experiment: LERAD

The experiments were performed on an 8th Gen Intel Core i5-8550U, 8GB SSD system. We use the following features for training to create the model:

- 'filename',
- 'parent_filename',
- 'parent_path_abs',
- 'path_abs',
- 'cert_signer',
- 'cert_trusted',
- 'privilege_level',
- 'parent_cert_signer',
- 'parent_cert_trusted',
- 'parent_privilege_level'

Further, we used following thresholds for rule generation based on multiple attempts at different threshold values.

Sample Size = 100,000
Max rules per pair = 50
Training/ Validation set % = .8 (80% of sample data for training and 20% for validation of candidate rules generated)

Candidate rule generation with sample size 100k, took 19,064.37 secs and generated 300 rules. The second pass is made for validation and rule pruning which took 33,492 secs and produced 232 validated rules that we would be using for detection.

The highest n/r (higher anomalousness) in the validated rule space was 28957 (n = 28957, r, =1) and the lowest n/r was 124.5 (n = 249, r =2) as shown below:

"IF parentfile_path_tr = svchost.exe->winsys+exe->SYSTEM
THEN parentfile_cert_tr = svchost.exe->Microsoft Windows->TRUE Score n = 28957 r = 1", and

“IF path_tr = pf+exe->pf+exe AND file_parent_tr = firefox.exe->firefox.exe AND parentfile_cert_tr = firefox.exe->Mozilla Corporation->TRUE AND parentfile_path_tr = firefox.exe->pf+exe->MEDIUM AND file_cert_tr = firefox.exe->Mozilla C orporation->TRUE THEN file_path_tr = firefox.exe->pf+exe->LOW or firefox.exe->pf+exe->MEDIUM Score n = 249 r = 2”

We used the model created to test on the malware execution logs, however, there were no rules that were broken by the malware logs. We further, tried using more/ less features, but that did not lead to more rules or flagging malware logs as malicious.

Post analysis of what is wrong with detection using LERAD, we believe that it makes too lazy rules because the fields are not strict. Further, the random pair selection in LERAD picks haphazardly a pair for candidate rule generation, however, in case there are no matching attributes, there are no rules generated, which perhaps is the reason why we don’t have more rules (as expected). LERAD with network traffic as used in the original paper [37] would generate more rules because with network traffic there are more common attributes shared by the pairs selected in the random selection (during initial rule making). However, because in host logs, the event data is not too strict, the rules are more loose, which we tried to solve using pairwise relations (as also used in the statistical model, we proposed), however, the rules were still not good enough for detecting the Lotl techniques. This brings us to the conclusion that we need to modify the random selection part of LERAD algorithm to use with host logs and produce more effective rules.

6.2 Experiment: Statistical Model

The experiments were performed on an 8th Gen Intel Core i5-8550U, 8GB SSD system. We use the following features to create the model. The following features are selected to create the relationship pairings in the data stream as shown in Table 2.

Features for model creation
lerad_id
filename
Parent_lerad_id
Parent_filename
Parent_path_abs
Path_abs
Cert_signer
Cert_trusted
Privilege_level
Parent_cert_signer
Parent_cert_trusted
Parent_privilege_level

Table 2: Features used for building model.

We build 15 pairwise relationships between the features for every parent-child process such as: parent filename → filename, parent filename + parent path → filename + path etc. relations that are used to build and add contextual information for each edge stream. These approximations are made using count min sketch with default parameters as $w = 1000000\ \ and\ d = 10$ (for our dataset, we observed no collisions using these default values)

We used 10 runs (to calculate the average timing) to build the model using count min sketch and it took ~1583 sec (averaged time) to compute the scores for all 15 pairwise relations.

We divide the dataset into 70+30, where 30% of the records are used for evaluating the performance of the learnt model and we look at the scoring we obtained for the malware logs that we simulated. We used the score of *0.1 as the anomaly threshold*, where values lower than that indicate that events are suspicious. There were approximately 17k anomalous events in the prediction set with the chosen threshold (0.1) as shown in Figure 10. This obviously has a huge number of FPs (even if the detection is good), hence, we used our proposed community detection method as described in Chapter 4, with default values for *depth >4 and mean community score > 0.2,* which provides ~220x less anomalous processes to look at. This produced 60 anomalous communities out of approximately 13k total communities created, as also can be seen in Figure 11.

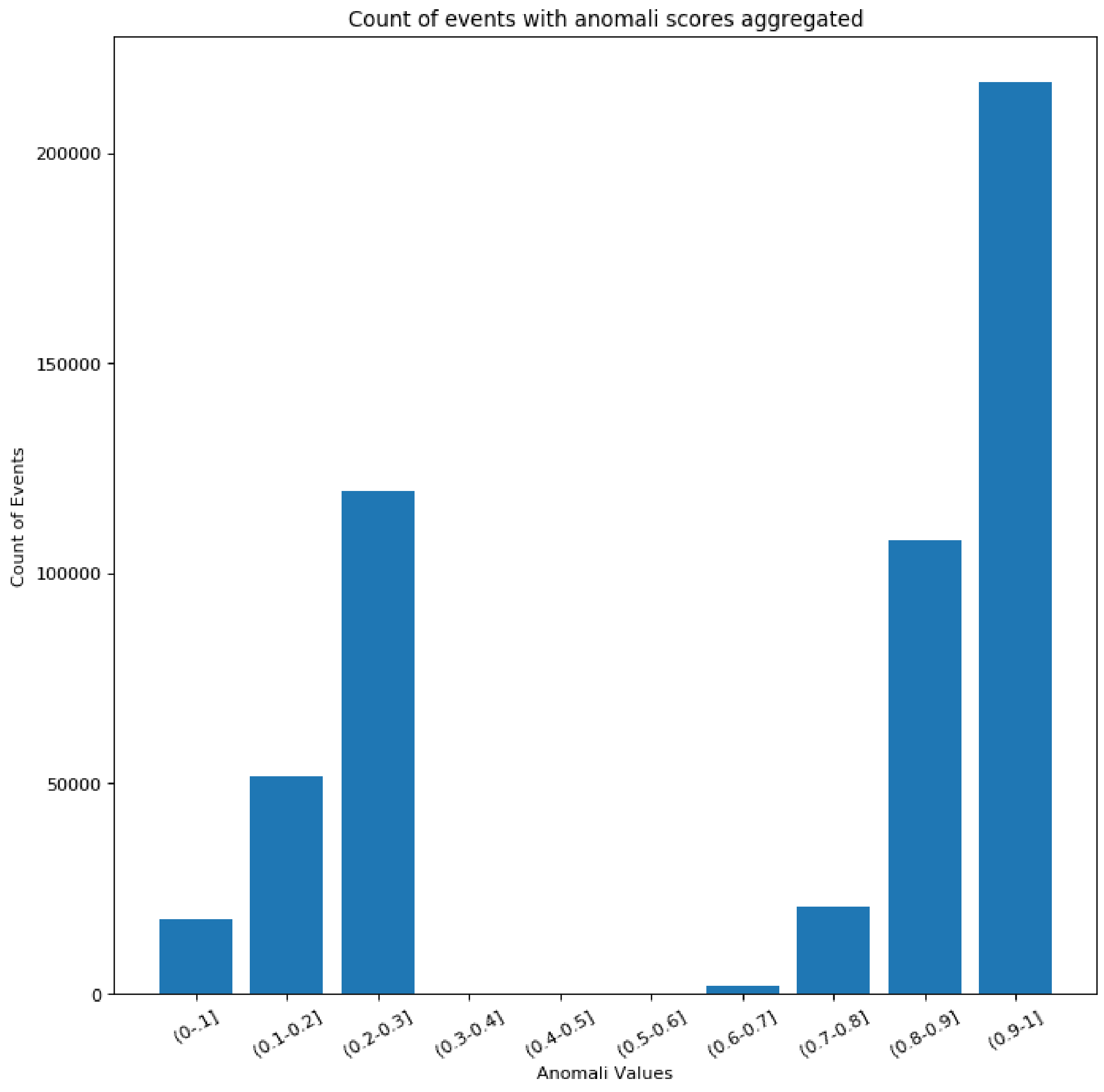


Figure 10: Count of events in prediction dataset with buckets

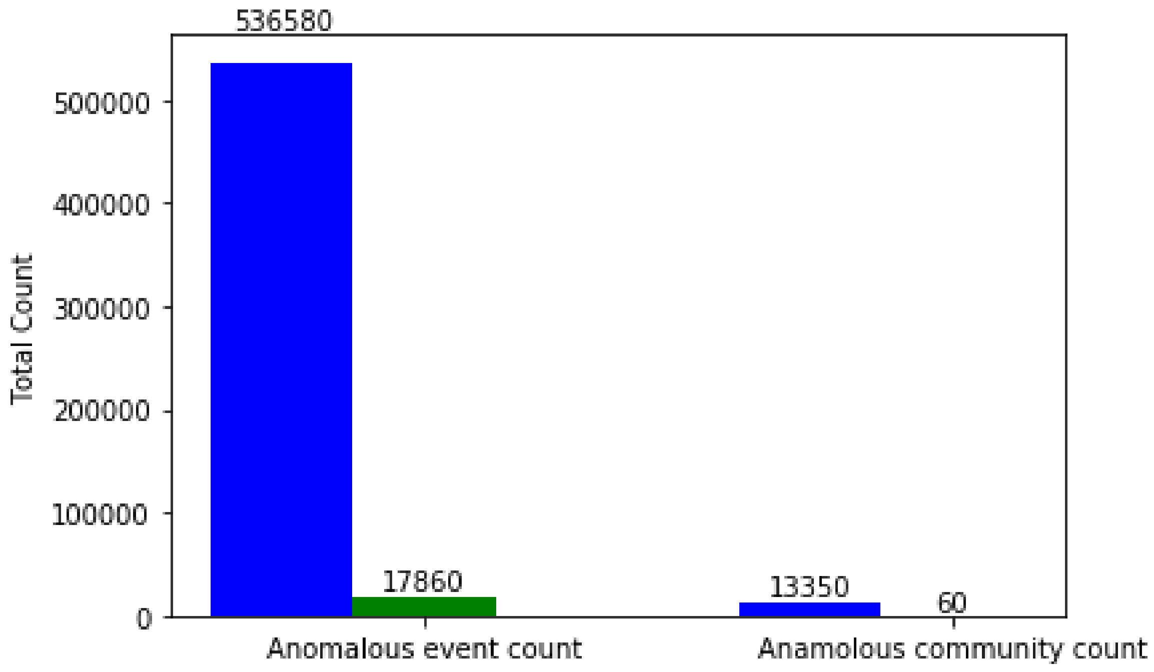


Figure 11: Reduction is overall anomaly space using community detection

Detection for Astaroth: The anomaly score generated by our proposed model puts the event from the Astaroth malware execution at bottom 0.2% of the prediction set. Higher score implies lower chances of being an anomaly and the event from malware received a score of 0.00631, which falls in the bottom 0.2%, thus indicating extremely high anomalousness. The highest score in the prediction set was reported to be .9683, implying least suspiciousness.

Further, using community detection, we observe that Astaroth produced 2 communities and both were in the 60 anomalous communities reported as seen in Figure 11. Details are in Table 3 below (Lower rank implies higher anomalousness).

Rank of Community	**Depth**	**Mean Relations broken**	**Mean Community Score**
4	6	18.5	0.0234
24	30	9.1	0.0725

Table 3: Detection Statistics for Astaroth

Detection for Azorult**:** The anomaly score generated by our proposed model puts the event from the Azorult malware execution at bottom 0.002% of the prediction set. Higher score implies lower chances of being an anomaly and the event from malware received the lowest score in the prediction set, score being 0.00356, which falls in the bottom 0.002%, thus indicating extremely high anomalousness. The highest score in the prediction set was reported to be .9683, implying least suspiciousness.

Further, using community detection, we observe that Azorult produced 1 community and was in the 60 anomalous communities reported as seen in Figure 11. Details are in Table 4 below.

Rank of Community	**Depth**	**Mean Relations broken**	**Mean Community Score**
4	34	16.2	0.0316

Table 4: Detection Statistics for Azorult

Detection for Dridex**:** The anomaly score generated by our proposed model puts the event from the Dridex malware execution at bottom 0.02% of the prediction set. Higher score implies lower chances of being an anomaly and the event from malware received the lowest score of 0.01034, which falls in the bottom 0.02%, thus indicating extremely high anomalousness. The highest score in the prediction set was reported to be .9683, implying least suspiciousness.

Further, using community detection, we observe that Dridex produced 1 community and was in the 60 anomalous communities reported as seen in Figure 11. Details are in Table 5 below.

Rank of Community	**Depth**	**Mean Relations broken**	**Mean Community Score**
9	7	13.5	0.127

Table 5: Detection Statistics for Dridex

Chapter 7

Conclusion

7.1 Attacks against the Detection Service

Keeping in mind that IDS is a key component of security layer for any organization, it is also necessary to look at how would it perform against adversarial attacks specially targeted against IDS. A detailed analysis on the adversarial attacks on IDS and a general taxonomy on the adversarial tactics is presented in paper by I. Corona et al. [69].

The paper lists out following main attacks goals against IDS:

Evasion and Response Hijacking: The intrusion pattern is modified in such a way that the system no more detects an intrusion and hence no alarms are raised. In hijacking, a pattern is crafted to generate an incorrect alert description and mislead IDS.

Overstimulation: The adversary mine patterns in logs that mimic the attacks patters but are benign and this causes poor performance in IDS by overwhelming them with high false positive rates.

Poisoning: This is rather complex technique, where the adversary creates attack patterns in the logs that are intended to corrupt the model in IDS systems that reply on historical data to learn the model, thus significantly hampering the accuracy of the IDS.

Denial of Service (DoS): The adversary overwhelms the IDS with increased traffic/ logs that overloads or slows down the IDS and pattern detection mechanism.

Reverse Engineering: The adversary gathers information about the internal detection processing of IDS, enabling the attacker to carry out any of the above attacks.

We would analyze our Detection System against the relevant attack tactics:

Corrupting the input data: The adversary can try to introduce errors or corrupt data from sensors employed by IDS. However, in our case the trust is placed on the callbacks from kernel itself and the adversary would need to attack the driver support routines (kernel-mode drivers) to corrupt the data. Further, there are custom kernel integrity checks placed to detect any such compromise that try to tamper OS kernel data structure.

Evasion and Overstimulation: An emerging technology called Generative Adversarial Network (GAN) that tries to attack any kind of machine learning systems using AI can be effectively used against our detection systems, since our detection learns based on the historic information generated from the host logs. Attacks generated by a GAN on machine learning systems act to confuse or fool the algorithm, and thus they can be used to poison the historic information or mine logs that mimic normal traffics and introduce high number of fresh contextual parent child relationships that would overwhelm the system with false positives.

For example, our first variant of rule-mining based IDS would be subjected to Exploratory Integrity attack that would focus on flooding the system using false negatives, allowing malicious traffic to enter the system. Fooling a sequence-based IDS involves sending a mass number of inputs, each with a slightly different intent, to try and get by all the rules set by the IDS. While in this thesis work we have not explicitly tested our detection service against attacks, however, we believe that the statistical variations introduced in the model can be used to detect when the learnt models differ significantly from the historic ones, however, this still assumes having the ground truth to build the initial models.

7.2 Summary

We have seen and evaluated two approaches in detecting Lotl techniques, where we were able to successfully detect with our proposed statistical method based on streaming data, however, with LERAD we were not able to produce rules that would detect the stealthy attacks. We also observed that using community detection approach, we were able to significantly reduce the FPs that a security analyst would have to look at while analyzing the detections results. The approach of looking at events as parent-child process enables us in building robust relational and graphical model to detect advanced techniques and tactics used by adversaries. The performance of our proposed work is decently fast enough, though, there is still possible improvements/ optimizations in speed using other languages such as C.

This thesis work helps us validate and cements the initial hypothesis that parent-child relationships can be used effectively in detecting novel attacks and identify anomalous parent-child anomalies. We believe our proposed work could be effectively used in domains of intrusion detection and fraud detection, specially where data can be visualized as graphs or hierarchical data. The approach can be used to incorporate contextual information and discover abnormal behavior as rare occurrences or outliers. Further, we understand and recognize that the proposed work has not been tested against more robust and varied datasets. That’s one of the limitation of the thesis work, since it hasn’t been tested against a large subset of attacks simply because it’s difficult and time consuming to stimulate behavior of malwares that use Lotl techniques and no publicly available dataset exists that can be transformed into parent-child relations to benchmark our findings. Possible future work in this direction would be to test against larger subset of malware attacks and to further improve on the reducing the FP rate using semi supervised setting in community detection.

References

[1] "Symantec, "Internet security threat report 2019"," [Online]. Available: https://docs.broadcom.com/doc/istr-24-2019-en.

[2] "Breach_LeveL_Index. (2017, November). Data breach statistics.," [Online]. Available: http://breachlevelindex.com/.

[3] "Preventing Attackers from Living off the Land," April 2019. [Online]. Available: https://www.edgewise.net/blog/preventing-attackers-from-living-off-the-land.

[4] "Living off the Land: Attackers Leverage Legitimate Tools for Malicious Ends," December 2019. [Online]. Available: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/living-land-legitimate-tools-malicious.

[5] "What Are Living Off the Land Attacks?," March 2020. [Online]. Available: https://logrhythm.com/blog/what-are-living-off-the-land-attacks/.

[6] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey. ACM Computing Surveys," *(CSUR), 41(3):15,,* 2009.

[7] X. Song, M. Wu, C. Jermaine, and and S. Ranka, "Conditional anomaly detection," *IEEETransactions on Knowledge and Data Engineering 19,5, 631–645,* 2007.

[8] A. Patcha, and and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *In: Computer networks 51.12 (2007). Elsevier, pp. 3448–3470..*

[9] T. Fawcett, and and F. Provost., "Adaptive fraud detection," *In: Data mining and knowledge discovery 1.3 (1997). Springer, pp. 291–316..*

[10] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and and M. Rajarajan., "A survey of intrusion detection techniques in cloud," *JNCA, 2013.*

[11] Kreibich C and C. J, "Honeycomb: creating intrusion detection signatures using honeypots," *SIGCOMM Comput Commun Rev 34(1):51–56,* 2014.

[12] I. Butun, S. D. Morgera, and and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE communications surveys & tutorials,* vol. 16, p. 266–282, 2014.

[13] R. Durst, T. Champion, B. Witten, E. Miller, and and L. Spagnuolo, "Testing and evaluating computer intrusion detection systems," *Communications of the ACM, 42(7),* pp. 53-61, 1999.

[14] M. Bhuyan, D. Bhattacharyya, and and J. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *, IEEE Communications Surveys & Tutorials 16,* vol. 1, pp. 303-336, 2014.

[15] I. Butun, S. D. Morgera, and and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE communications surveys & tutorials,* vol. 16, pp. 266-282, 2014.

[16] Khraisat et al, "“Survey of intrusion detection systems:techniques, datasets and challenges”," Vols. https://doi.org/10.1186/s42400-019-0038-7, 2019.

[17] N. Ye, SM Emran, Q. Chen, and and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection," *IEEE Trans Comput 51(7):,* p. 810–820, 2002.

[18] J. Viinikka, H. Debar, L. Mé, A. Lehikoinen, and and M. Tarvainen, "Processing intrusion detection alert aggregates with time series modeling," *Information Fusion,* vol. 10, pp. 312-324, 2009.

[19] W. Qingtao, and and S. Zhiqing, "Network anomaly detection using time series analysis,," *Joint international conference on autonomic and autonomous systems and international conference on networking and services - (icasisns'05),* pp. 42-42, 2005.

[20] I. Studnia, E. Alata, V. Nicomette, '. M. Kaâniche and Y. Laarouchi, "A language-based intrusion detection approach for automotive embedded networks," *Int J Embed Syst 10(1),* pp. 1-12, 2018.

[21] "Anomaly Detection – Another Challenge for Artificial Intelligence," [Online]. Available: https://www.experfy.com/blog/anomaly-detection-another-challenge-for-artificial-intelligence/.

[22] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance. Technical report,," *James P. Anderson Company, Fort Washington, Pennsylvania,* April, 1980.

[23] H. S. Javitz, A. Valdes, D. E. Denning, and and P. G. Neumann., "Analytical Techniques Development for a Statistical Intrusion Detection System (SIDS) Based on Account-ing Records. Technical report," *SRI International, Menlo Park, California,,* July 1986.

[24] T. F. Lunt, A. Tamaru, and and F. Gillham, "A real-time intrusion-detection expert system (IDES).," 1992.

[25] S. Chen, S. Cheung, R. Crawford, M. Dilger, J.Frank, J. Hoagl and e. al, "Grids—a graph based intrusion detection system for large networks," *Proceedings of the 19th national information systems security conference,* pp. 361-370, 1996.

[26] H. Debar and et.al, "A Neural Network Component for an Intrusion Detection System," *IEEE Computer Society Symposium on Research in Computer Security and Privacy. Oakland,,* pp. 240-250, 1992.

[27] W. LEE, S. J. STOLFO and K. W. ANDMOK, "A data mining framework for building intrusiondetection models," *Proceedings of the 1999 IEEE Symposium on Security and Privacy(Oakland, California,* May, 1999.

[28] F. Silveira, and and C. Diot, " URCA: Pulling out anomalies by their rootcauses," *Proc. IEEE INFOCOM,* pp. 1-9, March, 2010.

[29] S. Ranjan, S. Shah, A. Nucci, M. M. Munafò, R. L. Cruz, and and S. Muthukrishnan, "Dowitcher: Effective worm detection and containment in the Internet core," *Proc. IEEE INFOCOM, ,* pp. 2541-2545, 2007.

[30] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and and K. Cho, "Extractinghidden anomalies using sketch and non Gaussian multi resolution statistical detection procedures," *Proc. LSAD,* pp. 145-152., 2007.

[31] A. Lakhina, M. Crovella, and and C. Diot, "Diagnosing network-widetraffic anomalies," *Proc. ACM SIGCOMM,* pp. 219-320, 2004.

[32] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and and A. Lakhina, "Detection and identification of network anomaliesusing sketch subspaces," *Proc. 6th ACM SIGCOMM IMC,* pp. 147-152, 2006.

[33] W. Lee, and and S. J. Stolfo, "Data mining approaches for intrusion detection," *Proc. 7th USENIX Security Symp,* vol. 7, p. 6, 1998.

[34] R. Vaarandi, "Mining event logs with SLCT and LogHound," *Proc.IEEE NOMS,* pp. 1071-1074, April 2008.

[35] K. Yoshida, Y. Shomura and a. Y. Watanabe, "Visualizing networkstatus," *Proc. Int. Conf. Mach. Learning Cybern,* pp. 2094-2099, Aug 2007.

[36] V. Chandola, and and V. Kumar, "Summarization—Compressing data intoan informative representation," *Knowl. Inf. Syst.,* vol. 12, pp. 355-378, 2007.

[37] M.V.Mahoney, and and P. K. Chan, "Learning rules for anomaly detection of hostile network traffic," *Proc. 3rd IEEE ICDM,* pp. 601-604, 2003.

[38] L. Akoglu, H. Tong and D. Koutra, "Graph-based anomaly detection and description: a survey.," *Data Mining Knowl Discov 29(3),* no. doi:10.1007/s10618-014-0365-y, p. 626–688. , 2015.

[39] M. Salehi, and and L. Rashidi, "A survey on anomaly detection in evolving data: [with application to forest fire risk prediction]," *ACM SIGKDD Explorations Newsletter20(1),* pp. 13-23, 2018.

[40] Jimeng Sun, Dacheng Tao, and and C. Faloutsos, "Beyond streams and graphs:dynamic tensor analysis," *Proceedings of the 12th ACM International Conferenceon Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA,* p. 374–383, 2006.

[41] A. Beutel, W. Xu, V. Guruswami, C. Palow, and and C. Faloutsos, "Copy-Catch: stopping group attacks by spotting lockstep behavior in social networks," *Proceedings of the 22nd International Conference on World Wide Web,* pp. 119--130, 2013.

[42] Dhivya Eswaran, Christos Faloutsos, Sudipto Guha and a. N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," *Proceedings of the 24thACM SIGKDD International Conference on Knowledge Discovery &amp; DataMining. ACM,* p. 1378–1386, 2018.

[43] Minji Yoon, Bryan Hooi, Kijung Shin, and and C. Faloutsos, "Fast andaccurate anomaly detection in dynamic graphs with a two-pronged approach," *Proceedings of the 25th ACM SIGKDD International Conference on KnowledgeDiscovery & Data Mining,* p. 647–657, 2019.

[44] W. Yu, C. C.Aggarwal, S. Ma, and and H. Wang, "On anomalous hotspot discovery in graph streams.," 2013.

[45] K. Shin, B. Hooi, J. Kim, and and C. Faloutsos, "Densealert: Incremental dense-subtensor detection in tensor streams.," *KDD,* 2017.

[46] S. Ranshous, S. Harenberg, K. Sharma, and and N. F. Samatova, "A scalable approach for outlier detection in edge streams using sketch-based approximations," *Proceedings of the 2016 SIAM International Conference on Data Mining,* no. SIAM, p. 189–197, 2016.

[47] D. Eswaran, and and C. Faloutsos, "Sedanspot: Detecting anomalies in edge streams," *IEEE International Conference on Data Mining(ICDM),* p. 953–958, 2018 .

[48] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin and C. Faloutsos, "MIDAS: Microcluster-Based Detector of Anomalies in Edge Streams," *AAAI ,* 2020.

[49] Ming-Syan Chen, Jiawei Han and P.S.Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering,* vol. 8, no. ISSN: 1041-4347, pp. 866 - 883.

[50] R. Agrawal, T. Imieliński and A. Swami, "Mining association rules between sets of items in large databases," *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93,* p. 207, 1993.

[51] P. Hájek, I. Havel and M. Chytil, "The GUHA method of automatic hypotheses determination," *Computing. 1 (4): ,* p. 293–308, 1966.

[52] W. Cohen, "Fast Effective Rule Induction," *Proc. of the 12th Intl. Conf. on Machine Learning, Tahoe City, CA.,* pp. 115-123, 1995.

[53] R. Agrawal, and and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile,* pp. 487-499, 1994.

[54] G. Cormode, and and S. Muthukrishnan, "An improved data streamsummary: the count-min sketch and its applications," *J.Algorithms,* vol. 55, no. no. 1, pp. 58-75, 2005.

[55] G. Cormode, "Count-Min Sketch," *LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. ,* no. https://doi.org/10.1007/978-0-387-39940-9_87, 2009.

[56] R. Motwani, and and P. Raghavan, "Randomized Algorithms," *Cambridge University Press,* 1995.

[57] Gut and Allan, "Probability," *A Graduate Course (Second ed.). New York, NY: Springer,* no. ISBN 978-1-4614-4707-8, 2013.

[58] "Conditional probability distribution," [Online]. Available: https://en.wikipedia.org/wiki/Conditional_probability_distribution.

[59] He, Xiaofei, et and al., "Laplacian Score for Feature Selection," *NIPS,* 2005.

[60] "Feature Selection — Overview of Everything You Need to Know," [Online]. Available: https://mc.ai/feature-selection%E2%80%8A-%E2%80%8Aoverview-of-everything-you-need-to-know/.

[61] "PsSetCreateProcessNotifyRoutineEx function," [Online]. Available: https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/nf-ntddk-pssetcreateprocessnotifyroutineex.

[62] "MIT Lincoln Laboratory. (1999, June). DARPA Intrusion Detection Data Sets," [Online]. Available: https://www.ll.mit.edu/ideval/data/.

[63] M. Tavallaee, E. Bagheri, W. Lu, and and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *IEEE symposium on computational intelligence for security and defense applications,* pp. 1-6, 2009.

[64] G. Creech G and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguous and Discontiguous system call patterns," *IEEE Trans Comput 63(4),* p. 807–819, 2014b.

[65] [Online]. Available: https://mordordatasets.com/introduction.html.

[66] "Latest Astaroth living-off-the-land attacks are even more invisible but not less observable," [Online]. Available: https://www.microsoft.com/security/blog/2020/03/23/latest-astaroth-living-off-the-land-attacks-are-even-more-invisible-but-not-less-observable/.

[67] "Dridex: the secret in a PostMessage()," [Online]. Available: https://reaqta.com/2020/06/dridex-the-secret-in-a-postmessage/.

[68] "AZORULT Malware Information," [Online]. Available: https://success.trendmicro.com/solution/000146108-azorult-malware-information-kAJ4P000000kEK2WAM.

[69] I. Corona, G. Giacinto, and and F. Roli, "Adversarial attacks against intrusion de-tection systems: Taxonomy, solutions and open issues," *Information Sciences 239 (0) ,* p. 201 – 225, 2013.