Practical Intrusion Detection of Emerging Threats

Ryan Mills, *Member, IEEE,* Angelos K. Marnerides, *Member, IEEE,* Matthew Broadbent, *Member, IEEE,* and Nicholas Race, *Member, IEEE*

***Abstract*—The Internet of Things (IoT), in combination with advancements in Big Data, communications and networked systems, offers a positive impact across a range of sectors including health, energy, manufacturing and transport. By virtue of current business models adopted by manufacturers and ICT operators, IoT devices are deployed over various networked infrastructures with minimal security, opening up a range of new attack vectors. Conventional rule-based intrusion detection mechanisms used by network management solutions rely on pre-defined attack signatures and hence are unable to identify new attacks. In parallel, anomaly detection solutions tend to suffer from high false positive rates due to the limited statistical validation of ground truth data, which is used for profiling normal network behaviour. In this work we go beyond current solutions and leverage the coupling of anomaly detection and Cyber Threat Intelligence (CTI) with parallel processing for the profiling and detection of emerging cyber attacks. We demonstrate the design, implementation, and evaluation of *Citrus*: a novel intrusion detection framework which is adept at tackling emerging threats through the collection and labelling of live attack data by utilising diverse Internet vantage points in order to detect and classify malicious behaviour using graph-based metrics as well as a range of machine learning (ML) algorithms. Citrus considers the importance of ground truth data validation and its flexible software architecture enables both the real-time and offline profiling, detection and classification of emerging cyber-attacks under optimal computational costs. Thus, establishing it as a viable and practical solution for next generation network defence and resilience strategies.**

***Index Terms*—Intrusion Detection, Machine Learning, Threat Intelligence**

I. Introduction

THE advancement of technology has paved the way for the pervasive integration of computers into the lives of many. The data and resources they hold makes each a potential target for attackers. Disruption to these systems can incur financial losses for the operators, and wider consequences for users who are unable to access resources.

To defend against the growth in unique distributed infections, the method in which attacks are identified and prevented have received significant overhaul [1]. The defense mechanism of choice within modern enterprise networks is typically an Intrusion Detection System (IDS). An IDS performs classification of events within a network into either benign or malicious, in an attempt to thwart intrusion attempts. These events can be network based, such as packets or flows traversing a switch, or host based, such as process utilisation and system logs.

R. Mills, M. Broadbent, and N. Race is with the School of Computing and Communications, Lancaster University, Lancaster, Lancashire, LA1 4YW, England, United Kingdom (e-mail: r.mills2@lancaster.ac.uk, m.broadbent@lancaster.ac.uk, n.race@lancaster.ac.uk)

A. Marnerides is with the School of Computing Science at the University of Glasgow, Lilybank Gardens, G12 8RZ, Scotland, United Kingdom (e-mail: angelos.marnerides@glasgow.ac.uk)

The decision making process can be categorized as either *misuse* or *anomaly* detection. Misuse detection uses predefined attack patterns using signatures of known malware. As a result, novel attacks exploiting unknown vulnerabilities bypass this approach as no corresponding signature exists [2]. Misuse detection relies upon a database of known attack signatures, which is necessarily large and requires constant updates to keep abreast of developing threats. Anomaly detection techniques leverage a description of normal behaviour which is learned through observations in training data, and any future observation which deviates from the normal baseline is labeled as malicious. Consequently, attacks which were not previously encountered are still possible to detect. Nonetheless, the task of profiling normal behaviour is highly challenging due to its dependency on the establishment of ground truth data which in many cases are not validated either statistically or even empirically. Hence, anomaly detection solutions can also be problematic especially when large volume of information is processed, analysed, and statistical normality is not thoroughly assessed.

The requirements distilled by the nature of the emerging cyber threat landscape demonstrate that attacks are now conducted on a large scale, thus IDS-based solutions need to maintain vast quantities of either signatures or training data used for decision making. Therefore a high throughput processing framework is required to adequately analyse the data in a reasonable time. Recently, novel frameworks have emerged which possess the ability to handle the large amount of data required for contemporary intrusion detection. Providing anomaly detection algorithms the ability to scale with the influx of emerging threats is crucial to overcome the challenges brought about by the rapid growth of connected devices, innovative infection techniques, and large volumes of data [3].

When using supervised statistical algorithms to determine abnormality in systems telemetry, it is essential to train the models on data which includes benign and relevant attack behaviour. There exist challenges with contemporary intrusion detection data sets, including containing dated and non-relevant attacks [4], and manually injected attacks which do not accurately reflect the current threat landscape [5]. Honeypot telemetry provides improved training data that better represents the emerging threat landscape [1], as the data contained within is a partial view of the malicious actions currently propagating throughout the Internet. Furthermore, one of the most crucial features in supervised machine learning algorithms is is the ground truth represented as target labels. Current data sets do not either include this feature [6], or engineer it in a non-standard and often problematic manner [7], [8].

The main contributions within this paper are:

1) **Practical integration of Cyber Threat Intelligence (CTI) for active network defense**: In contrast to many commercial solutions where CTI is a passive process involving the offline composition of normal traffic behaviour, Citrus is the first to utilise real-time CTI feeds for ground truth data validation such as to aid towards accurate online anomaly detection.
2) **Real-time anomaly detection:** We demonstrate that Citrus provides a robust data pipeline by exploiting properties of data and system parallelism satisfying the data processing requirements for real-time anomaly detection. Thus, acting as a viable approach for next generation network defense solutions.
3) **Attack data availability:** Due to the challenges identified with current data sets during literature review, Citrus orchestrates the collection, processing, and analysis of emerging threat data through the composition of honeypots scattered throughout Lancaster University's public address space. Citrus then produces an open and reusable labelled data set suitable not only for the evaluation of itself, but also for the evaluation of next generation intrusion detection techniques. The constant nature of the proposed honeypot telemetry scheme and labelling approach enables this data set to be updated. Thus, critically reflecting the various novel threats encountered in the wild which enables an evolving understanding of malicious behaviour. To the best of our knowledge, an updated data set which incorporates attacks captured by honeypots has not previously been achieved.

The remainder of this paper is structured as follows: Section II provides an overview of background and related work whereas Section III presents the motivating factors behind Citrus' design choices. Section IV outlines the architecture of Citrus, highlighting the frameworks which are integrated. Section V presents the environment in which experiments are conducted and the data set used to evaluated Citrus. Section VI describes the various methodologies employed by Citrus and scenarios designed to evaluate them. Section VII provides analysis of the results obtained through these scenarios. Finally, Section VIII concludes the paper.

II. Background and Related Work

A. Modern Threat Landscape

Networked computer systems are increasingly susceptible to abuse due to the growing propagation of exploitation and disruption campaigns orchestrated throughout the Internet. Motivated by economic and geopolitical gain, these myriad attacks encompass a large range of techniques and are very diverse in nature. Ranging from IoT (Internet of Things) malware initiating large scale DDoS attacks, to targeted and stealthy intrusions into critical infrastructure leveraging zero-day exploits as part of a sophisticated attack chain, the emerging threat landscape remains a challenging problem to solve.

Multi-stage intrusions are a special type of attack which incorporate multiple techniques, tactics and protocols to achieve a predetermined goal. In this scenario, attackers typically follow an attack life cycle model which details the steps required by an adversary to infiltrate computer systems. Several of the stages inherent within this type of attack are used in Section VII to evaluate Citrus' capacity to detect emerging threats.

B. Cyber Threat Intelligence

As identified in the previous section, the evolving threat landscape consists of innovative and large scale attacks. The monitoring of these attacks is essential to reduce the rate at which they propagate, a process which utilises sensors scattered around the Internet to observe bleeding edge adversarial methodology. The information derived from these sources is known as Cyber Threat Intelligence (CTI), which in contemporary studies has been used to profile malicious activity and enhance security mechanisms.

1) Honeypots: Honeypots are systems under observation which contain components that masquerade as legitimate enterprise infrastructure in order to catch unsuspecting adversaries leveraging previously unobserved exploits, attack tactics and patterns used for infiltration [9]. Utilising these honeypots grants unrestricted access to emerging CTI through extracted log data, which is otherwise extremely difficult for the wider community to access due to corporations limiting the exposure of breaches. Therefore, honeypots are an invaluable asset in the identification of novel attack vectors. The nature of the deployed honeypots dictate the types of attacks encountered, with honeypots deploying a large number of different services granting observation of more diverse types of attacks and corresponding CTI data. Yahya et al. [10] deploy a range of emerging honeypots which emulate a large amount of popular service. Through the analysis of the captured telemetry, the authors identify diverse attack techniques such as DDoS attacks, bruteforce attempts, and exploitation of vulnerabilities targeting various software implementations.

2) Cyber Threat Intelligence Services: CTI services provide organisations insight into relevant threat actors which aim to infiltrate infrastructure. The greatest majority of CTI services function in a bespoke manner, and deliver various types of intelligence. These range from services which systemically scan and profile the entire internet address range, e.g. Shodan[1], to services which maintain historical records of identified attackers in blacklists such as Maltiverse[2].

Another such service which performs profiling of the Internet is Greynoise[3]. This service claims to maintain a large number of shifting servers in hundreds of data centers across the world, and perform omni-directional scanning to uncover properties about the topology of the Internet. Due to commercial reasons, the true nature behind how these services operate is not known to the general public. Despite this, the intelligence offered by these platforms is an invaluable resource in the analysis and defense of emerging threats.

[1] https://www.shodan.io/
[2] https://maltiverse.com/
[3] https://greynoise.com/

C. Intrusion Detection

An intrusion can be defined as any activity that causes damage to information systems [11]. An IDS is a system which identifies these malicious actions to ensure the security and integrity of computer systems is maintained. These systems monitor various activity taking place, and if deemed to be malicious, alerts and remediation procedures are commenced to mitigate such threats. Flow-based Network-based IDS (NIDS) methods have become popular in literature as they encompass measurements from multiple packets in a connection and only analyse the packet header, not the payload. This is beneficial since full payload capture is computationally expensive and can be lead to performance bottlenecks in high-speed networks [12].

1) Misuse Detection: IDSs can be categorised based upon the method used to identify attacks. These groupings are known in literature as misuse based intrusion detection and anomaly based intrusion detection. Misuse based IDSs typically leverage a database of known attack signatures to detect attacks. These systems use pattern matching to identify whether suspect activity is contained within a known malicious database.

They also generally grant excellent detection accuracy for known attacks. However, they struggle against emerging and novel threats, due to the fact that the signature database must be updated to reflect the new attacks.

2) Anomaly Detection: Anomaly detection approaches have drawn immense interest from the research community due to them alleviating challenges incurred by signature based systems. Most notably, these include the ability to detect zero-day attacks as they do not require a known malicious signature database. In this approach, a model of normal systems behaviour is created using machine learning, statistical-based or knowledge-based methods [11]. Any monitored activity which deviates from this normal behaviour profile is treated as an intrusion. However, these systems generally suffer from a greater false positive rate compared to signature alternatives. This is caused through various means, including a lack of strong ground truth within training data.

In literature, machine learning based methods have been shown to be adept in the detection of a variety of malicious activity. In this approach, machine learning models are trained using data extracted from intrusion detection data sets. There are many distinct types of machine learning algorithms that have been applied to intrusion detection scenarios, such as decision trees, clustering, neural networks, and genetic algorithms. For example, Sangkatsanee et al. leverage decision trees to detect a variety of malicious behaviour, including DDoS and probes, in an online fashion [13]. This approach verifies the detection capabilities through real attacks orchestrated against victims, where the network traffic is captured and accurately classified. However, this approach does not consider attacks which are sophisticated and emerging in nature.

3) Intrusion Detection Data Sets: In order to evaluate the effectiveness of an IDS, the accuracy of the attack detection procedure is typically used. This process typically assesses the FP (False Positive) and TP (True Positive) rates of detection. A *data set* which contains both benign and attack traces is necessary to assess this metric. Currently, there still exists challenges in the collection and release of this data to the public. Recent studies have shown the most widely adopted data sets are heavily dated and no longer reflect modern attack vectors. Moreover, investigations have also discovered other deficiencies relating to lack of traffic diversity, attack techniques, and inappropriate features which can affect the transparency of the IDS evaluation [14].

In the security community there is a lack of public IDS data sets, with current literature urgently appealing for high quality labelled attack data [15], [16], [7], [3], [17], [18]. An IDS data set of high quality must include a comprehensive reflection of contemporary threats and a range of benign behaviour spanning multiple protocols, hosts, and applications [14]. Furthermore, an often overlooked aspect is the methodology behind labelling of the data set. Correct labelling ensures all observed attacks are identified, thus dictating the reliable outcome of any IDS [19]. Additionally, labelling enables the use of supervised machine learning algorithms, which have been shown to usually provide better detection accuracy than their unsupervised alternatives [20]. A study conducted by Abt et al. discovered that the majority of publicly available data sets are not labelled, assigning the cause to be due to the labour intensive nature of manual labelling of data sets [8]. The authors then argue that the *"missing labelled data problem"* affects repeatability and comparability of research.

The KDD '99 data set is one of the first publicly available IDS data sets and remains the most widely adopted in the evaluation of anomaly detection methods [16]. The data set incorporates results from a simulation containing both normal and abnormal traffic collected from inside a military network.

As identified by McHugh, no attempt was made to ensure that the injected synthetic attacks were realistically distributed in the background noise [21]. Furthermore, Tavallaee et al. claims an inherent problem within the data set is that the workload of the synthesized data does not reflect traffic in real networks [22]. These problems highlight the challenges incurred when utilising synthetic data in the generation of IDS data sets, heralding innovation in order to capture a high quality realistic data set. However, the main problem with this data set resides within the fact that it is extremely dated, with the attacks no longer being relevant to what is experienced today.

The Kyoto 2006+ data set was created by researchers at National Institute of Information and Communications Technology through the collection of attacks and benign telemetry spanning from November 2006 until August 2009 [4]. In their approach, Song et al. utilise honeypots as an attack capture medium, deploying various enticing systems which lure adversaries into revealing their techniques. In the same network, servers conducting benign tasks such as mailing service and DNS server were also deployed to generate realistic background traffic resembling legitimate enterprise infrastructure.

Data Set Labelling In order to successfully label the data set, the *ground truth* must be discovered. In literature, there are only a handful on publicly available data sets which consider

Name	Labelling Mechanism	Open Source	External Correlation	Data Set Released	Attack Types
Citrus	CTI	Yes	Yes	Yes	Various
MAWI [23]	Unsupervised AD	No	No	Yes	Various
Sperotto et al. [24]	Log Correlation	No	No	No	Various
Aparicio-Navarro et al. [25]	Dempster-Shafer	No	No	No	Wireless
B-IDS [26]	SVM, RPCL, and Dempster-Shafer	No	No	No	Various

TABLE I: Comparison of Citrus and other frameworks which develop a ground truth.

the ground truth and provide corresponding labels. The KDD '99[4] ground truth is an example of a data set labelled through manual anaylsis.

Since the public release of KDD '99, novel approaches which provide automated routines to derive ground truth for intrusion detection data sets have been outlined in literature. One of the first data sets to incorporate such an approach was the Kyoto 2006+ data set [4]. In their approach, honeypots were deployed to attract malicious intrusion attempts. Kyoto 2006+ provides data with partial labels such as to enable the separation of normal and malicious traffic. Unlike the KDD data set, the labels do not describe explicit types of attacks. Much attention has been given to enhancing classification performance using data with partial labels, such as [27], [28], due to the less effort required to provide such labels. This is also true in network security, since specifying the distinct attack description is often more challenging. In this case, Kyoto 2006+ provides labels by assigning traffic relating to honeypots deployed within their architecture as malicious. Consequently, this approach does not consider legitimate traffic involving honeypots.

As identified by Sperotto et al. in [24], not all telemetry associated with honeypots are attacks and malicious in nature. In their work, they identify a number of non-malicious traffic to and from honeypots which they consider "side effects". More recently, it has also been identified that honeypots are also targeted by legitimate services which actively probe and document Internet connected devices, such as Shodan [29]. In order to differentiate between actual attacks and normal traffic relating to honeypots, a strong ground truth is required.

The approach taken by B-IDS [26] leverages one-class Support Vector Machine (SVM) and a Rival Penalized Competitive Learning (RPCL) network to develop attack ground truth. These algorithms produce a classification outcome which is combined with the Dempster-Shafer theory such as to produce a singular output. Via this approach, the authors are able to distinguish between normal and attack packets extracted from real traffic traces with an error rate of around 2%.

A comparison of Citrus with other frameworks which provide automated ground truth development routines are listed in Table I. In addition, we provide an experimental comparison with Citrus and B-IDS in Section VII-D to further illustrate Citrus' beneficial properties.

[4]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

III. Motivation

A. Lack of Emerging Attack Telemetry

During the process of data set creation, it is essential to identify certain properties that ensure it can be successfully leveraged by the research community. Małowidzki et al. outline several of these facets in [17]. Realism and a holistic ground truth are among the features which compose a high quality data set according to Małowidzki et al. In order for an intrusion detection data set to be as realistic as possible, there are multiple considerations which need to be made. These include the composition of network traffic captured from actual networks instead of simulated network traffic through mathematical models [30]. Furthermore, attack telemetry which is representative of real attacks currently propagating should also be incorporated. The research conducted by Hindy et al. surveys the available IDS data sets, and finds that data sets with dated attacks render IDS ineffective against emerging attacks or zero days [1].

The nature of honeypots provides an unrestricted view of malicious activity currently propagating over the Internet, capturing detailed accounts of attacks likely currently being encountered by enterprise networks. As a result, honeypots have recently been identified as a solution to the problems faced by data sets requiring realistic malicious activity [3]. Motivated by similar approaches leveraging honeypots to capture a wide variety of contemporary malicious behaviour (e.g., [4], [24]), such as brute force, worms, and exploits, we use attack data collected by the deployment of honeypots to compile a new intrusion detection data set.

B. Significance of Robust Class Labels

Correctly labelling the data set ensures all attacks are successfully identified, separating the benign traces from the malicious traces. When taking a supervised ML approach, these systems require labelled data sets to be trained. Moreover, in more general scenarios the real nature of data set must be known in order to evaluate the effectiveness of the detection approach, i.e. to determine the TP/FP rates of detection.

Within typical network conditions, collecting labelled data sets is impossible. At present, the majority of data sets are labelled manually by a technician performing forensic analysis, which is impractical especially when the amount of data is large, taking considerable amounts of time, thus not enabling on-line implementation. In the case of the KDD data set, labels were verified by hand thus resulting to an extremely labour intensive process [5].

In order to successfully and accurately implement an automatic labelling method, a robust ground truth must be derived.

A novel approach to labelling is explored in the design of Citrus, through the correlation with third party CTI services a fuller understanding of how actors conduct their behaviour with regards to the general Internet is deducted. Through this active correlation with diverse sources, the real nature of the data is discovered, further enhancing the accuracy of real-time anomaly detection algorithms for network defense purposes.

C. Importance of Real-Time Detection

Intrusion detection frameworks must operate in an online fashion to provide network administrators timely alerts for mitigation and remediation purposes. In modern literature, a great number of implemented systems perform batch processing and offline classification of network telemetry [31], [32]. In these cases, timely remediation is not possible since the detection process occurs much later than the actual malicious behaviour. These types of publications typically attempt to marginally increase detection rates on outdated data sets, which, as identified in [8], is a current problem facing the research community.

A practical, online approach to intrusion detection ensures that attacks are able to be detected in a realistic networked environment. To achieve this type of intrusion detection, telemetry must be transmitted to Citrus where it is then classified in real-time to determine whether malicious actions have taken place. Recent investigations have revealed that existing approaches to anomaly detection are not effective enough, especially upon the consideration of real-time prediction [33]. This is due to the requirement of vast training data and the sheer amount of data within modern networks which needs to be assessed. Therefore, in our approach, we leverage parallelism through a cluster of processor nodes in order to alleviate the aforementioned challenges.

IV. Citrus Architecture

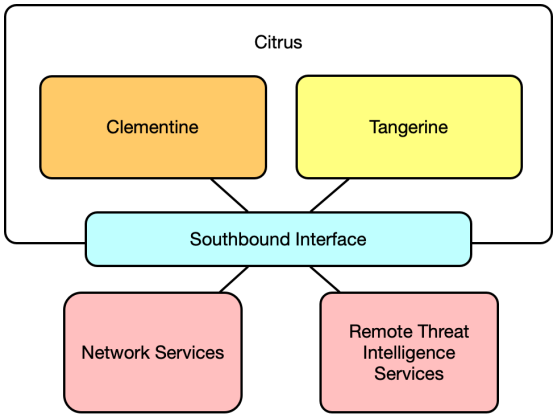

Fig. 1: The design of Citrus' architecture.

As illustrated in Figure 1, the Citrus architecture[5] is composed of distinct components which interface with services deployed within the network, as well as remote services located on the Internet. The southernmost components within Figure 1 represent these services which provide Citrus crucial input data necessary for its operation. Furthermore, they are also utilised for output operations, such as saving labelled telemetry to disk for future dissemination within the research community.

The northernmost components represent the two modules implemented to aid CTI gathering and real time anomaly detection. Clementine is a component within Citrus which rapidly identifies malicious behaviour occurring within the local network through the utilisation of machine learning models. The Tangerine component within Citrus performs automatic intrusion detection data set labelling through correlation with CTI service providers.

A. Southbound Interface

The Southbound Interface is composed of various modules that are used to communicate with services located within the network and on the Internet.

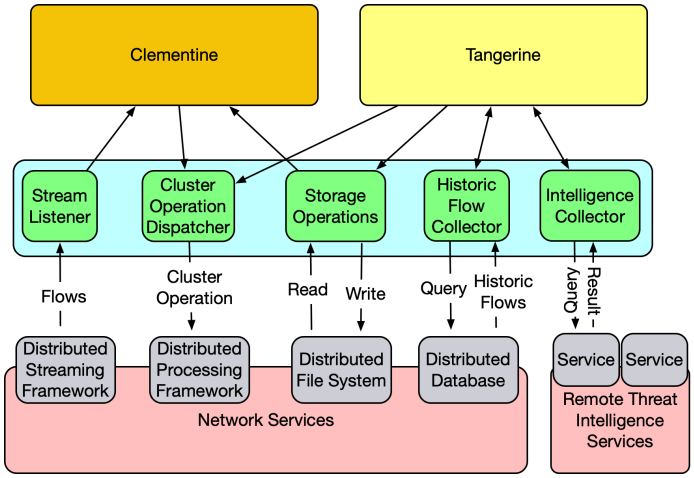

Fig. 2: Overview of the design of the Southbound Interface.

1) Stream Listener: This module solely serves as an input to Clementine. Clementine uses the *Stream Listener* input to gather local telemetry, and perform online flow-based intrusion detection. As such, the information flowing through it consists of network flows emanating from devices located within the network.

This is achieved through integration with Apache Spark's streaming library. Citrus utilises Apache Spark[6] to underpin it's large scale data processing capabilities. The Apache Spark interface enables structured data processing, machine learning and structured streaming for incremental computation and stream processing.

Spark Streaming extends the traditional core Spark API by providing high-throughput, fault tolerant, and scalable stream processing of live data. This data can emanate from a variety of sources. The streaming platform implemented within Citrus is Apache Kafka[7]. This platform was chosen due to its distributed, replicated, and extremely highly performant nature.

[5]https://github.com/ruzzzzz/Citrus

[6]https://spark.apache.org

[7]https://kafka.apache.org

Spark Streaming additionally enables the application of machine learning algorithms upon such data streams. Spark provide a high-level abstraction of these streams of data, called Discretized Streams (DStreams). DStreams are internally represented of sequences of RDDs (Resilient Distributed data sets). RDDs represent a collection of elements which can be operated on in parallel. The sequencing of these elements structure a streaming computation as a series of micro-batch computations. This property is leveraged by Citrus to perform online intrusion detection using machine learning algorithms on batches of data with small time intervals.

2) Cluster Operation Dispatcher: The *Cluster Operation Dispatcher* provides an interface between Citrus and distributed processing frameworks. This module utilises clusters of executor nodes deployed within the network to perform parallel computation of complex operations. This integration enables highly efficient transformations of vast, high-dimensional intrusion detection data sets. As mentioned previously, Citrus utilises high level abstractions from Spark to achieve this.

3) Storage Operations: The *Storage Operations* module is responsible for the reading and writing of labelled data sets and trained models to a distributed file system. As a result, this also allows Clementine to access labelled telemetry in an distributed fashion, decreasing loading time as records are partitioned between executor nodes. Hadoop File System[8] (HDFS) was chosen as the storage medium due to it's high-throughput access and fault tolerant nature.

4) Historic Flow Collector: The *Historic Flow Collector* module provides an interface between Tangerine and a distributed database. The distributed database stores historic flow-based telemetry emanating from honeypots, in which attack traffic occurs, and services deployed privately within the network, in which benign traffic occurs. Critically, this module enables Tangerine to read vast amounts of telemetry from a database, which will then be used to compile a robust intrusion detection data set.

5) Intelligence Collector: Unlike the other modules discussed in this section, the *Intelligence Collector* module provides an interface between Citrus and remote services located on the Internet. Through this module, Tangerine is able to correlate the data derived from honeypot deployments with various CTI services to gain contextual information regarding suspect attackers. The CTI services leveraged by Tangerine are heterogeneous in nature and provide varying information. As illustrated within Figure 3, the *Intelligence Collector* module provides both input and output operations. This is to enable custom queries (as an output) to each distinct CTI service, such as providing a list of suspicious hosts, and returning to Tangerine (as in input) the corresponding response from the service.

B. Tangerine

This component ultimately outputs a comprehensive flow-based labelled data set, which is then utilised by the *Clementine* module to perform online intrusion detection tasks. To achieve this functionality, *Tangerine* initially gathers the telemetry to be labelled from the Southbound Interface. The telemetry is then pre-processed in preparation for exportation as a labelled data set. Contextual information is then gathered through correlation with diverse CTI services.

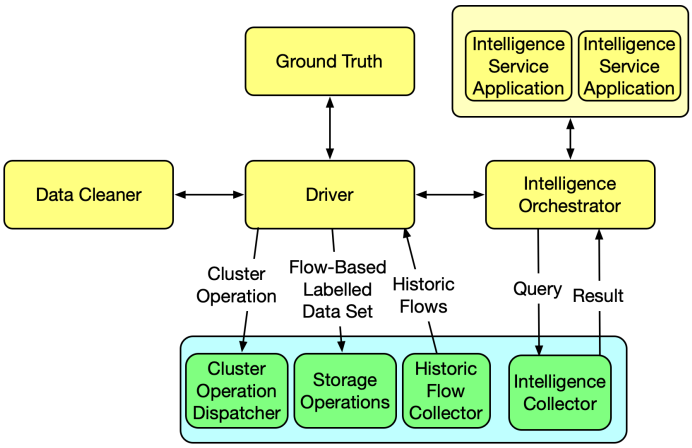

Fig. 3: Overview of the design of Tangerine.

1) Driver: The *Driver* component is the entry point of Tangerine, and is additionally responsible for the orchestration of all stages within the labelling process. The *Driver* receives, as an input, flow based telemetry collected from honeypots and benign servers deployed within the network, which is delivered through the Southbound Interface.

Upon the telemetry being successfully cleaned, contextualised, and labelled, the *Driver* is able to produce a flow-based labelled intrusion detection data set. This output is destined for storage within a distributed file system.

2) Data Cleaner: The *Data Cleaner* component is responsible for the transformation of telemetry into an appropriate format. All telemetry gathered from honeypots and benign servers deployed within the network is processed through this component to prepare the relevant features required for an intrusion detection data set. As a result, the *Data Cleaner* plays a critical role in delivering the telemetry in a format appropriate for the evaluation of Intrusion Detection Systems.

Hence, this component receives from the Driver raw telemetry in the form of network flow measurements. The *Data Cleaner* transforms the raw telemetry and returns it to the Driver in a standard format, preparing it to be output as a data set.

3) Intelligence Orchestrator: The *Intelligence Orchestrator* component coordinates requests for Cyber Threat Intelligence (CTI) data. It achieves this through the instantiation and governing of *Intelligence Service Applications*, custom applications built specifically for communication with a certain CTI service. As such, this component plays a critical role in enabling extensibility and flexibility for future integration with new and evolving CTI services. Upon the receipt of a list of hosts extracted from raw network telemetry by the Driver, this module begins its operation. Consequently, when a successful response is received from the Southbound Interface, the *Intelligence Orchestrator* then instructs each *Intelligence Service Application* to parse the response.

[8]https://hadoop.apache.org/

Name	Entity Type	Description
Apility	Blacklist	Provides the capability to query IP addresses against a diverse range of blacklists.
Censys	Service	Search engine for Internet connected devices.
GreyNoise	Service	Search engine for Internet connected devices.
Hybrid Analysis	Malware, C&C Servers	Scans uploaded samples and provides access to several extracted IoCs.
Maltiverse	Blacklist	Provides querying of multiple blacklists.
OTX	Blacklist	Provides access to custom blacklist by Alienvault.
Shodan	Service	Search engine for Internet connected devices.
ZoomEye	Service	Search engine for Internet connected devices.

TABLE II: The CTI services used to correlate data points and provide a ground truth.

4) Intelligence Service Application: *Intelligence Service Applications* are developed within Tangerine which contain the functionality required to query and parse responses from CTI services. As these services are heterogeneous in nature, there is no standard method in which to construct a query and parse the corresponding response. As a result, to enable communication between Tangerine and specific CTI service, a corresponding application must be instrumented which contains this distinct functionality. These applications provide their procedures to the Intelligence Orchestrator which further coordinates the intelligence gathering process.

The Intelligence Service Applications are implemented in a manner which promotes extensibility, and currently support a diverse range of CTI services. These are further detailed in Table II.

5) Ground Truth: The *Ground Truth* component is responsible for the identification of supernodes inherent within a graphical relationship of suspected attackers. These supernodes represent nodes which are highly connected to malicious entities. The nodes within the graph are deduced through the active CTI correlation performed by the Intelligence Service Applications discussed above. Hence, intelligence is passed to this module by the driver in the form of dictionaries on a daily basis.

By virtue of various integrated libraries, this module contains the ability to create graphs and perform clustering. In detail, these libraries include networkx[9] and scikit-learn[10] Python libraries. Networkx provides support for graph data structures including directed graphs and multigraphs, as well as graph algorithms such as PageRank. The scikit-learn library is used for the provided k-means algorithm.

Based upon this approach for supernode identification, the telemetry associated with each node within the graph can be labelled. Upon the successful calculation of graph features and identification of supernodes, the *Ground Truth* component provides a list of labels to the Driver.

C. Clementine

The *Clementine* component within Citrus, as illustrated in Figure 4, performs online intrusion detection of flows emanating from devices within the network. These network flows are transmitted to *Clementine* through the Southbound Interface using streaming frameworks. This component adopts a machine learning approach to intrusion detection. In this approach, supervised classification models are trained on the labelled flow-based data set output by Tangerine. Both the initial training data set and the streaming flows are input to Clementine through the Southbound Interface.

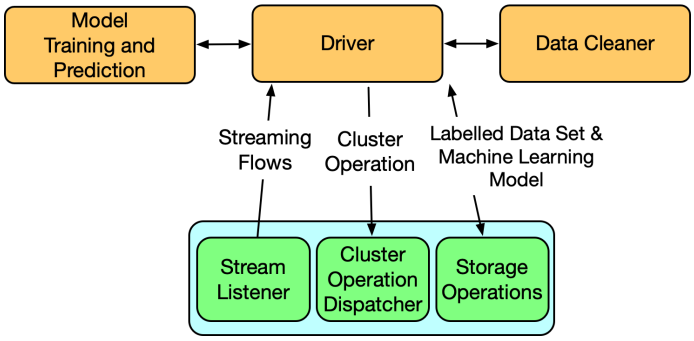

Fig. 4: Overview of the design of Clementine.

As motivated by contributions within the research community, *Clementine* integrates with Spark's machine learning library (MLLib) which performs efficient computation of machine learning algorithms on distributed nodes within a cluster.

1) Driver: The *Driver* within Clementine fundamentally coordinates all stages within the intrusion detection process. Initially, this component receives, as an input, a flow-based labelled data set from the Southbound Interface. It subsequently instructs the Model Training and Prediction component to build a trained classification model from this data set. Upon successful training of the model, Clementine is ready to begin predicting whether streaming flows emanating from devices within the network are of a benign or malicious nature.

To begin this operation, the *Driver* starts to receive and process network flows transmitted through the Southbound Interface. Every network flow is then sent to the Data Cleaner module in the form of an RDD. Relevant features are then prepared from the raw telemetry and returned for further classification. Instructions are then given to the Model Training and Prediction component to predict the label associated with the flow.

2) Model Training and Prediction: The *Model Training and Prediction* component provides the machine learning capabilities used by Clementine to perform Intrusion Detection. This component performs two critical tasks in the intrusion detection process: the training of a machine learning model on a flow-based data set containing both benign and malicious traces, and the prediction of streaming flows emanating from devices within the network. Both of these tasks require features

[9]https://networkx.github.com
[10]https://scikit-learn.org/stable/

of flows to be sent from the Driver. Upon the receipt of these features, training and prediction operations are executed on a cluster of worker nodes.

This is implemented through leveraging high level abstractions provided by Spark. A critical abstraction leveraged by this module is the ML Pipeline. The ML pipeline specifies a ML workflow through the chaining of *Transformer* and *Estimator* algorithms. A *Transformer* is an abstraction which converts one partitioned collection of elements to another, typically used to append columns such as feature vectors. An *Estimator* abstracts learning algorithms which train on data. This abstraction ultimately produces a model. The ML Pipeline enables the appropriate sequencing of stages required to perform online intrusion detection.

V. Evaluation Environment & Dataset Description

A. Evaluation Environment

In order to evaluate Citrus, a suitable network environment is required. A research facility located within Lancaster University, the Cyber Threat Lab [34], is used for this purpose. Consisting of multiple inter-connected components, the Cyber Threat Lab grants access to a plethora of malicious data garnered by a large variety of sources. This facility is physically composed of five servers, each containing an Intel Xeon E5-2620 v3 processor and 128GB RAM. The VMWare[11] hypervisor is used to manage virtual machines within this environment.

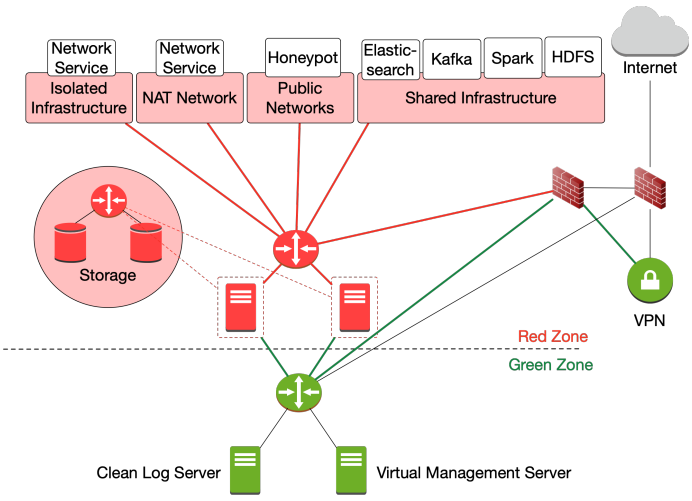

Fig. 5: The architecture of the Cyber Threat Lab.

Citrus requires various services to be deployed within the network. A range of these services are instantiated by the authors within the Cyber Threat Lab which support Citrus and the work in this paper in general. Notably, Citrus requires telemetry which incorporates malicious and benign activity. A variety of different honeypots are deployed which emulate various vulnerable services, including the recently released TPot[12]. TPot is composed of a number of medium interaction container based versions of popular honeypots which span multiple protocols. Consisting of over fifteen honeypots in total, TPot enables the capture of a wide variety of emerging attack telemetry, including attacks targetting IoT and ICS devices.

The telemetry associated with every deployed honeypot is captured by Cisco's Joy[13] tool. Joy provides a libpcap[14] based software solution for the extraction of data features from live network traffic. This is achieved using a flow oriented model, and is represented as JSON. This telemetry is transferred to a distributed database located within the Shared Infrastructure network segment. The database deployed within the Cyber Threat Lab is Elasticsearch[15]. Elasticsearch also provides a convenient agent, Logstash[16], which is used to transfer all telemetry from the honeypots to a database instance.

As one of the requirements of Citrus is to perform intrusion detection by leveraging machine learning algorithms, it is also necessary to capture benign telemetry. This provides a profile of benign behaviour, which is used by such algorithms to correctly identify flows which do not pose a threat. This known benign traffic is captured from internally accessible VMs located within the Shared Infrastructure. In a similar vein to the approach taken by Song et al. [4], these VMs perform two main pieces of functionality, acting as a DNS server and a data node. Fundamentally, all traffic related to these network services are regarded as benign as there were no observed attacks within this controlled environment. A Spark cluster consisting of four nodes is also provisioned with 4 vCPU cores and 32GB RAM.

B. Data Set

The telemetry captured and labelled using Citrus is compiled to create a flow-based intrusion detection data set with a robust ground truth. In this section, the properties of all telemetry captured within the operational period is presented.

The operational period, in which automatic network telemetry collection and labelling is conducted, initiated in June 2020. Due to the automatic nature of this process, there is no fixed end date. As a result, the intrusion detection data set compiled from this telemetry will receive periodical updates for the foreseeable future. However, the analysis performed in this paper uses data captured until October 2020. This novel intrusion detection data set is named LUFlow '20. LUFlow '20 is released to the general public through a GitHub repository[17]. This release anonymises IP addresses to alleviate privacy concerns.

	Number of flows	Mean flows per day
Total	101,116,515	849,718
Benign	53,921,369	453,120
Malicious	36,810,141	309,328
Outlier	10,385,005	87,268

TABLE III: Distribution of flow labels within LUFlow '20.

[11] https://vmware.com
[12] https://github.com/dtag-dev-sec/tpotce
[13] https://github.com/cisco/joy/
[14] https://www.tcpdump.org/
[15] https://elastic.co
[16] https://www.elastic.co/logstash
[17] https://github.com/ruzzzzz/LUFlow

1) Dataset Overview: During this current period of observation, there has been a total of 101,116,515 flows captured within the Cyber Threat Lab. Of which, 53,921,369 flows are known to be benign. As previously discussed, we regard all traffic relating to privately accessible internal network services as benign. The telemetry captured through the composition of honeypots within the Cyber Threat Lab is subject to Tangerine's labelling mechanism. The number of malicious flows labelled in this manner is 36,810,141. The nodes identified as having no association with malicious entities through this correlation process, have their corresponding telemetry labelled as an outlier. The total count of these outlier flows within LUFlow '20 is 10,385,005. These flows remain in the data set to encourage the practice of manual analysis to determine the true intent behind the unsolicited form of communication.

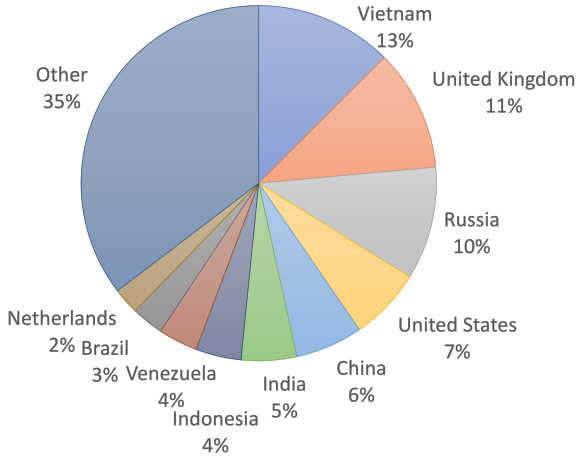

Fig. 6: Geolocation distribution of unsolicited traffic.

IP Address	Flow Count	ASN	Country
x.x.x.1	423,777	37963	CN
x.x.x.2	397,955	57678	RU
x.x.x.3	302,137	49877	RU
x.x.x.4	279,090	213371	NL
x.x.x.5	243,113	199264	ET
x.x.x.6	179,851	213371	NL
x.x.x.7	176,472	49877	RU
x.x.x.8	174,738	208666	ET
x.x.x.9	165,861	49877	RU
x.x.x.10	147,132	49877	RU

TABLE IV: The top 10 source IP addresses within '20.

2) Geolocation Analysis: The results outlined in Table IV showcase the locations of the top 10 source IP addresses identified in the telemetry captured by nodes within the Cyber Threat Lab. The IP addresses of these servers have been anonymised to consider the privacy of the individuals. Notably, AS49877 appears in four separate instances within this table. This AS is associated with a hosting provider serving the Russian and Moldovan regions.

The geographic distribution of all flows associated with the deployed honeypots, i.e. malicious or outlier flows, is illustrated in Figure 6. In total, there are flows associated with 188 distinct countries within LUFlow '20. In this investigation, it is observed that flows relating to servers originating in Vietnam are the most prevalent. It is also observed that around 50% of all unsolicited flows captured by honeypot telemetry originate from five countries: China, Vietnam, United Kingdom, United States, and Russia.

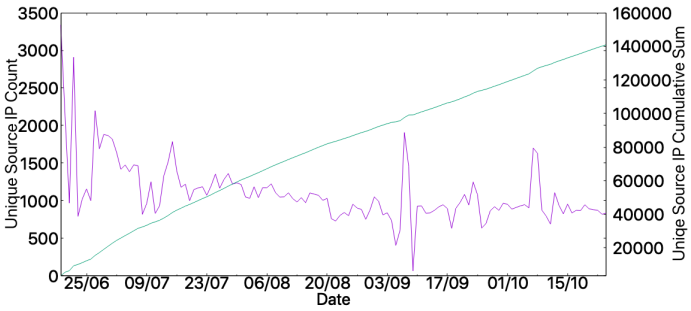

Fig. 7: Distinct count of source IP addresses per date.

3) Source IP Address Analysis: Figure 7 illustrates statistics regarding the number of source IP addresses identified within the telemetry. The purple line represent the occurrence of unique source IP addresses in each day, while the green line represents the accumulation, i.e. the cumulative sum, of these unique IP addresses identified. This overall count is steadily increasing, with an average number of 1,110 new source IP addresses being discovered every day.

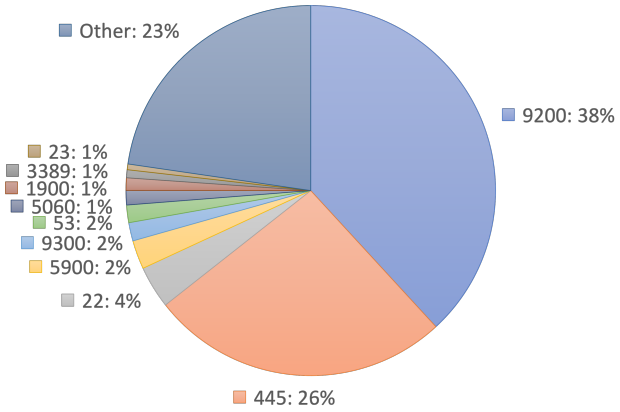

Fig. 8: Distribution of destination ports.

4) Destination Port Analysis: Figure 8 depicts the various destination ports which have been used to communicate with services within the Cyber Threat Lab. These include services which have been targeted by attackers, as well as network services are used to profile benign behaviour. The LUFlow '20 data set contains flows which explore every available port, ranging from 0 to 65535. As evidenced in the figure, the destination port 9200 occurs the most commonly within LUFlow '20. This port is frequently used to communicate with the distributed database which stores the telemetry used to compile LUFlow '20, and as a result is mainly used for benign purposes.

The next most prevalent port within the data set is 445, which is typically used by SMB services. This service has

#	Name	Description
1	src_ip	The source IP address associated with the flow. This feature is anonymised to the corresponding Autonomous System.
2	src_port	The source port number associated with the flow.
3	dest_ip	The destination IP address associated with the flow. The feature is also anonymised in the same manner as before.
4	dest_port	The destination port number associated with the flow.
5	protocol	The protocol number associated with the flow. For example TCP is 6.
6	bytes_in	The number of bytes transmitted from source to destination.
7	bytes_out	The number of bytes transmitted from destination to source.
8	num_pkts_in	The packet count from source to destination.
9	num_pkts_out	The packet count from destination to source.
10	entropy	The entropy in bits per byte of the data fields within the flow. This number ranges from 0 to 8.
11	total_entropy	The total entropy in bytes over all of the bytes in the data fields of the flow.
12	mean_ipt	The mean of the inter-packet arrival times of the flow.
13	time_start	The start time of the flow in seconds since the epoch.
14	time_end	The end time of the flow in seconds since the epoch.
15	duration	The flow duration time, with microsecond precision.
16	label	The label of the flow, as decided by Tangerine. Either benign, outlier, or malicious.

TABLE V: The LUFlow '20 inherent feature set.

recently received patches which aim to fix critical vulnerabilities, such as RCE (Remote Code Execution). This investigation has identified a number of these vulnerabilities which are still being actively exploited in the wild. Most notably, the infamous Eternalblue (CVE-2017-0144 and CVE-2017-0145) exploit which caused devastating damage through the incorporation into malware and botnets such as WannaCry and Petya. Due to the powerful nature of these exploits, they are still propagating at an alarming rate, as evidenced by the high number of requests to this port within LUFlow '20.

5) Extracted Features: As mentioned previously, each row within the LUFlow '20 data set represents a distinct network flow captured within the Cyber Threat Lab. Each column within the data set represents a feature which describes a certain facet of the flow. Inspired by predecessor intrusion detection data sets, LUFlow '20 contains a variety of significant features extracted from both benign and malicious flows. These features, which are recorded in Table V, incorporate both packet-based and flow-based features. Each flow is defined as a set a packets with common characteristics. In this instance, the conventional network five-tuple is used: source IP address, source port number, destination IP address, destination port number, and protocol. Furthermore, bidirectional flows are created by combining unidirectional flows which are part of the same session. Bidirectional flows consist of a pair of unidirectional flows whose source addresses, destination addresses and ports are reversed. This enables both inbound and outbound communication within a single flow. Critically, LUFlow '20 provides a ground truth through flow labels. Since all attacks are real, i.e. they are not injected into the data set, it is impossible to accurately label each *type* of attack. Therefore, the target labels of each flow are considered to be either benign, outlier, or malicious.

6) Data Set Comparison: A comparative analysis is conducted between LUFlow '20 and other related IDS data sets surveyed in recent literature. Only publicly available data sets which incorporate a ground truth in the form of target labels are considered in this comparison. Table VI documents the properties of each these data sets.

In summary, the key differences which separates LUFlow '20 from the majority of other IDS data sets is the *real* nature of network traffic and incorporated attacks which reflect emerging threats currently propagating. As detailed in Section IV, the design of Citrus ensures *real* benign and malicious traffic are constantly captured, labelled, and published in periodic releases of LUFlow '20. The majority of data sets surveyed in literature are created using simulation tools, which fundamentally generate synthetic network traces. Additionally, LUFlow '20 does not contain any attacks which have been manually injected into the data set. This ensures that all of the malicious activity incorporated within is truly representative of emerging attack vectors.

Furthermore, LUFlow '20 is the only *recent* data set which receives constant updates. As documented in Table VI, the MAWILab data set is updated in daily intervals. However, the labelling mechanism was implemented based upon the output of the combination of four outdated unsupervised learning algorithms. During the period between the detector implementation and present day, there has been a substantial evolution in the way in which attacks manifest themselves, which suggests this approach is no longer relevant. As other researchers have indicated, this approach lacks accuracy for modern day attack vectors and network traffic in general [38].

Fundamentally, LUFlow '20 was created with the intention to constantly capture network traces incorporating real attacks and benign behaviour with a robust ground truth. The constant nature of this telemetry capture enables the data set to be updated. Critically, this allows LUFlow '20 to reflect novel threats encountered in the wild, making the creation and release of future intrusion data sets which aim to capture modern threat vectors redundant.

VI. Methodology

A. Establishing Ground Truth

Based upon the restricted ground truth identified in literature, we have decided to place an emphasis on the creation of a robust ground truth. Citrus' Ground Truth module contains the functionality required to map any identified relationships relating to remote servers which interact with deployed hon-

Name	No. of networks	No. of distinct IPs	Simulation	Attack Injection	Duration	Updated
KDD '99 [5]	2	11	Yes	Yes	5 Weeks	No
MAWILab [23]	1	Unspecified	No	No	19+ Years	Yes
Kyoto 2006+ [4]	5	4,420,971	No	No	2 Years	No
ISCX 2012 [35]	4	21	Yes	Yes	7 Days	No
CTU-13 [36]	2	Unspecified	No	Yes	6 Days	No
UNSW-NB15 [14]	3	45	Yes	Yes	16 Hours	No
CICIDS2017 [37]	5	500	Yes	Yes	1 Week	No
LUFlow '20	4	132,177	No	No	4+ Months	Yes

TABLE VI: Comparison of IDS data sets.

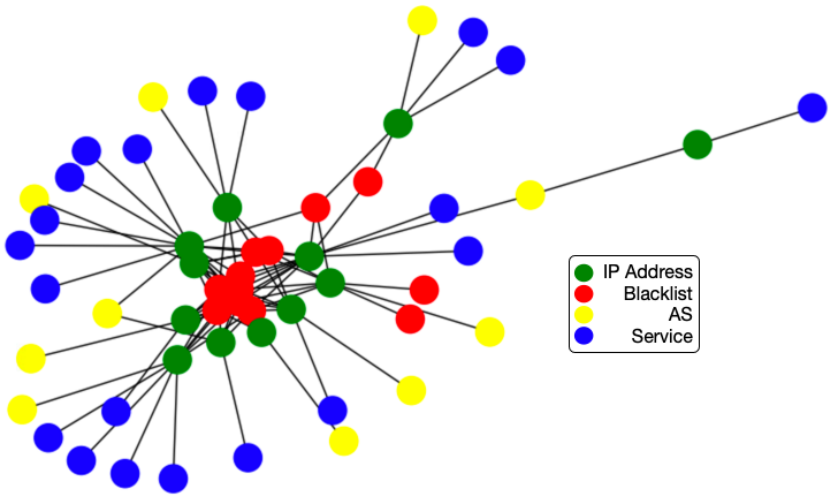

Fig. 9: Subgraph example of node relationships.

eypots in the Cyber Threat Lab. These relationships are then used to derive the ground truth.

Initially, the raw flow measurements are collected, and the unique IP addresses are extracted. CTI services are then queried, using the IP addresses as parameters. The data collected from these services is then processed, and each record' s date is compared against the date the captured telemetry occurred. Records with matching dates are then extracted and used for plotting within a graph. This process ensures derived relationships are valid for the suspected attacker on the date they interact with deployed honeypots.

This network of relationships derived from third party CTI services is defined as a simple undirected graph, which is composed of nodes connected by edges. Formally, we define the graph as $G = (V, E)$, where V is the set of vertices, and E is the set of edges.

In this instance, the nodes within the graph are the suspected attackers IP addresses, which are connected through edges to entities derived from the collected CTI. These connected entities represent all known associations of a suspect attacker present on the date of the telemetry capture. Therefore, this method identifies nodes which have been observed to be performing malicious actions on the date the telemetry was captured.

Due to the diversity and heterogeneity resulted by the variety of CTI services utilised by Tangerine, each service provides an edge between a suspect attacker, u, and a variable entity. These entities currently represent blacklists, v, malware samples, w, Autonomous Systems, x, (through ASNs), and available services, y. Considering this, we define the set of vertices in G as:

$$V = \{u_1, ..u_n, v_1, ..v_n, w_1, ..w_n, x_1, ..x_n, y_1, ..y_n\} \quad (1)$$

and there exists an edge from u to v, w, x, or y if the CTI collected indicates a connection between them.

Figure 9 illustrates a small sub graph used to highlight the approach. Blacklists, denoted by red entities, reside in the centre and are connected to the IP addresses, green nodes, which are active within the blacklist on the date of telemetry capture. These nodes are then also connected to an ASN, yellow entities, and exposed services, blue entities. In order to identify nodes which belong to a large number of malicious entities, such as blacklists, features are extracted from the graph.

The features used in this approach include node degrees and eigenvector centrality. For a particular node in a graph, degrees represents the total number of connected edges. High values of degrees indicate a large number of relationships to entities. Formally, the maximum degree of a vertex can be defined by $deg(v) = n - 1 \ \forall \ v \in G$. The rationale behind using this feature is that if a node is connected to a large number of entities, the likelihood is that it has been

identified as performing many malicious actions over the Internet. Eigenvector centrality is the measure of influence a node has in a graph. A high eigenvector centrality value means that the node in question is connected to many nodes who themselves have a large number of connections. This feature is used to understand how important a particular node is within the graph as it gives a clear indication of how connected a node is, both directly and indirectly. For the graph, G, let $A = (a_{v,t})$ be the adjacency matrix where

$$a_{v,t} = \begin{cases} 1 \; if \; vertex \; v \; is \; linked \; to \; vertex \; t \\ 0 \; if \; vertex \; v \; is \; not \; linked \; to \; vertex \; t \end{cases} \quad (2)$$

The eigenvector centrality score for a given vertex, v, can then be given as

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} a_{v,t} x_t \quad (3)$$

where $M(v)$ is the set of neighbours of vertex v and λ is a constant.

These features are then input to the k-means clustering algorithm that we utilise in this work. The k-means algorithm is an unsupervised method which partitions observations into k clusters, in which each observation belongs to a cluster with the nearest centroid.

A number of clusters within the graph identify nodes which are the most highly connected to the entities derived through CTI collection. Such clusters include potential attacker nodes which have a very high number of connections to blacklists and malware samples on the date the telemetry was captured, and as such they can be treated as supernodes. Any node in the graph which is not linked to a supernode is labelled as an outlier. These outliers can be further examined to deduce the intent behind their communication with a honeypot. The remaining nodes within the graph that are linked to a supernode, i.e. there exists an edge, or a series of edges, between the two nodes, are labelled as malicious. This is due to the fact that they share a common entity association. To find every vertex which is linked to a supernode, a breadth first search is performed. This procedure has a time complexity of $O(|V| + |E|)$ for each vertex, since all vertices must be explored in the worst case.

This approach forms the basis of the labelling of honeypot telemetry. Figure 10 presents a full graph of all node relationships for a certain date as derived through correlation with CTI services. The nodes which represent IP addresses extracted from the telemetry are coloured based upon the aforementioned labelling approach. The nodes labelled as malicious are coloured red, as they are connected to a supernode, and outlier nodes are coloured green. The cyan nodes indicate an entity as derived through CTI.

Validation of the consistency within these clusters of data is required to ensure the identified supernodes, and derived ground truth, are accurate. The silhouette metric is used for this purpose. The silhouette metric shows how similar an object is to its own cluster, compared to other clusters. Silhouette measurements range from -1 to +1, where a high value indicates that the object is very similar to objects its own cluster, and not similar to objects in other clusters. For any data point i within the cluster C_i let

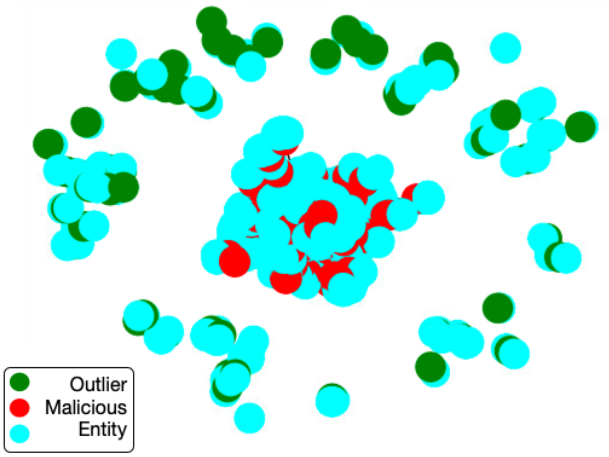

Fig. 10: Graphical representation of nodes distinguished by label.

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \quad (4)$$

be the mean intra-cluster distance, where $d(i,j)$ is the distance between i and j in the cluster C_i. The mean dissimilarity between i and a cluster, C_k, in which i is not a member can be defined by

$$b(i) = \max_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_i, i \neq j} d(i,j) \quad (5)$$

The silhouette coefficient of i can now be defined by

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, if \; |C_i| > 1 \quad (6)$$

These measurements for each data point, i, can be displayed visually by combining the coefficients into a single plot, enabling an appreciation of the overall quality of the clusters. Therefore, the average silhouette value, i.e. width of the plot, provides an evaluation of clustering validity [39].

B. Detection Performance

The detection performance is demonstrated through leveraging the proof-of-concept implementation, Clementine. Two experiments are conducted which evaluate Clementine's capacity to detect a variety of emerging attacks using a distributed processing approach. The performance of each of these algorithms are assessed by comparing the predicted label to the actual label. This assessment forms a confusion matrix which describes all possible classification outcomes.

In both of these experiments, Clementine is installed on a VM, which is provisioned with 4 vCPU cores and 32GB RAM, in an isolated network within the Cyber Threat Lab. Furthermore, the flows labelled as outliers are removed from the data set in each experiment, enabling a binary classification outcome. This is because the true intent of the flow is not

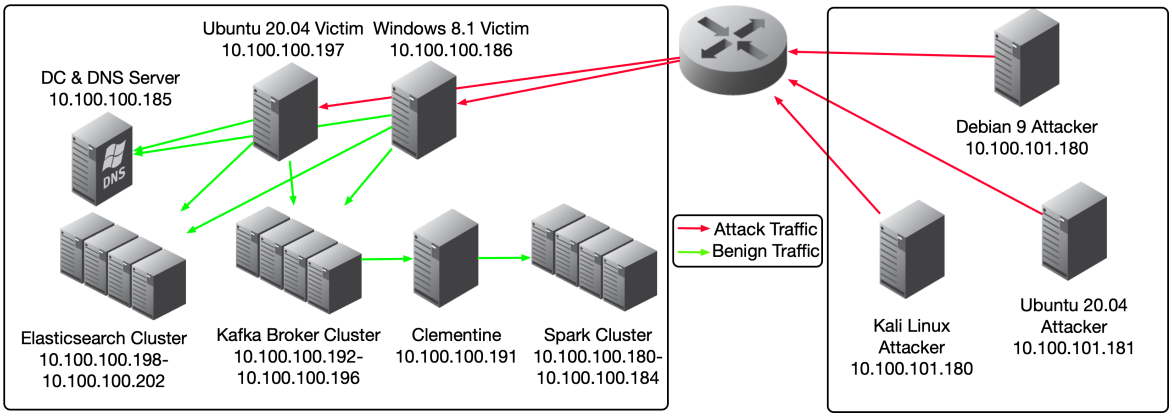

Fig. 11: Configuration of experimental set-up.

known. In the experiments performed in this evaluation, a positive prediction is where Clementine produces a label of *malicious*. Therefore, a True Positive (TP) refers to a prediction of *malicious* when in fact the flow in question is *malicious*, else it is treated as a False Positive (FP). On the contrary, a negative result occurs when Clementine predicts a *benign* label. As a result, if the flow under scrutiny is not related to attack traffic and Clementine predicts a *benign* label, a True Negative (TN) occurs. Furthermore, if Clementine predicts a *benign* label when the flow is related to malicious behaviour, a False Negative (FN) occurs.

In the first experiment, an offline detection approach is taken which considers various classification methods. In the second experiment, online intrusion detection is performed using live network data, which incorporates injected attacks, and the classification method which performed the best in the first experiment. The online evaluation is performed to assess Clementine's effectiveness in malicious behaviour detection using a realistic network environment.

1) Offline Detection: This experiment evaluates the capacity of machine learning algorithms to detect malicious activity within LUFlow '20. A myriad of traditional supervised learning algorithms are considered and evaluated within this section. This experiment is intended to verify the validity of the data set whilst identifying the greatest performing algorithm to use in the online detection process.

In this experiment, the LUFlow '20 data set is split into a training and test data set. The training data set consists of 30,000,000 randomly sampled records in total, with 15,000,000 benign records and 15,000,000 malicious records. The test data set contains 8,000,000 records, split into 4,000,000 malicious and 4,000,000 benign classes.

2) Online Detection: In this experiment, the gradient boosted tree classifier is used, as it performed the best during offline detection, to detect malicious behaviour in an online fashion using live network telemetry. The data used to train the model consisted of 30,000,000 malicious and benign records respectively. After having successfully trained the classifier on this training data, Clementine is ready to begin the consumption of live network telemetry in order to perform online intrusion detection.

For the purpose of this experiment, additional VMs are provisioned within the Cyber Threat Lab. An illustration of how these are connected within the network to detect real attacks is included in Figure 11. A victim server is instantiated within the shared infrastructure to allow intrusion attempts from attack servers located in the other networks. The victim server performs similar functionality to the benign network services used to profile benign behaviour within LUFlow '20, including coordination with a distributed database amongst other known benign behaviour. The attack servers attempt to infiltrate the victim server through various means.

As this telemetry within this evaluation is generated irrespective of Tangerine, the labelling process is not the same. Due to the prior knowledge about the injected attacks, flows are labelled based upon the IP addresses associated with the flow. If either the destination or source IP address correspond to an attack server, the flow is labelled as malicious, otherwise benign.

In this evaluation, a series of separate experiments are conducted which examine Clementine's ability to detect emerging threats in an online fashion using live network telemetry. Network telemetry is captured and streamed to Clementine over a period of 10 minutes. At a random point during this period, the benign telemetry is injected with malicious behaviour through attacks performed by various attack servers. The various scenarios used in this evaluation are documented below.

Scanning: In this scenario, a single attacker host performs a port scan of the victim machine. The Nmap tool is used to perform such network scan. Specifically, TCP SYN and UDP scanning techniques are leveraged to identify thousands of open ports upon the victim machine.

DDoS: This attack involves flooding the victim machine with a large amount of TCP-SYN requests to overwhelm its resources. This is orchestrated by numerous attack machines located within the same network. These machines initiate flooding for a period of 2 minutes. This is achieved through leveraging the hping3 tool.

Brute Force: The use of brute force and dictionary attacks plague networked systems to this day. Sending rapid amounts of authentication requests, these attacks can be orchestrated against remote or local targets to gain a foothold into infrastructure and gather sensitive information. In this experiment,

an MSSQL server is targeted with a large amount of brute force traffic for a period of 2 minutes. Hydra is used to perform this attack, further utilising the largely popular 'rockyou.txt' word list as the attempted passwords.

Multi-stage Intrusion: In this experiment, an attack is leveraged against a victim machine which consists of a number of distinct stages. Initially, the EternalRomance exploit (CVE-2017-0145) is utilised to gain remote code execution privileges on the victim machine. The payload is composed of shell code which instantiates a reverse shell on the victim. This reverse shell provides a covert method of interaction with the victim, and is further used to perform various actions, such as downloading sensitive documents to model data exfiltration in this experiment. These separate stages are designed to emulate emerging attack patterns encountered in real-world scenarios.

As the EternalRomance exploit leverages Microsoft's implementation of the SMB protocol, a Windows 8.1 VM was instantiated within the Cyber Threat Lab to play the role of the victim. A reverse shell payload is generated using the msfvenom framework. The exploit is then executed against the victim, gaining access to the aforementioned reverse shell through execution of the post-exploitation payload.

C. Classification Performance

1) Model Training: Training machine learning models enables inference and classification of future unknown records. Each of the machine learning algorithm associated with these models typically exposes tunable hyperparameters to developers which adjusts how the algorithm performs internally. For example, the Radial Basis Function (RBF) kernel in Support Vector Machine (SVM) algorithms enables the specification of C and γ parameters. It is not known beforehand which C and γ are optimal for a given problem, therefore, a *grid search* must be performed. The goal of this process is to identify the values for these parameters which increase prediction accuracy of unknown data.

In typical embodiments of grid search functionality, a separate model is trained *sequentially* for each parameter value combination specified in the grid search. The best performing model, i.e. the most accurate, is then selected for future prediction purposes. The Spark engine provides an alternative implementation which leverages the power of cluster computation. In this approach, separate models can be trained in parallel within executors on nodes in the cluster.

In order to evaluate the performance benefits of this approach, an experiment is conducted which compares grid search implementations. The training data used in this experiment is 500,000 randomly sampled records extracted from the LUFlow '20 data set.

Measurements are recorded for the total time taken to train all models, as specified by the parameters, and find the best performing upon a cluster of varying node sizes. In total, 4 Spark executor nodes, each containing 4 processor cores, are used to provide a maximum of 16 cores. The number of nodes in the cluster is used as the independent variable, and is changed to determine how it affects the total computation time. The mean of five separate measurements for each node size is taken to record an average total training time.

2) Streaming: As previously discussed, the Clementine module within Citrus integrates with Spark's DStream abstraction and performs intrusion detection in an online, practical fashion to detect emerging threats. DStreams enable the receipt of live input data streams and divides the data into batches, where it is then processed. The time taken to process and classify unknown records is of great importance, and should be as little as possible to rapidly notify systems administrators that malicious actions have taken place.

This evaluation aims to assess the speed at which a large amount of data can be streamed, processed, and classified using the proof-of-concept implementation, Clementine. The interval of the batches determine the rate at which data is processed. Therefore, careful consideration should be taken when choosing this value. The batch interval is used as independent variable in this evaluation to determine an optimal value in this instance.

Data containing 100,000 randomly sampled records extracted from LUFlow '20 is streamed to Clementine, where it is then pre-processed and classified, using the gradient boosted tree classifier, into malicious or benign classes. Five separate measurements are taken for every batch interval value to derive the average time taken to process all streamed records.

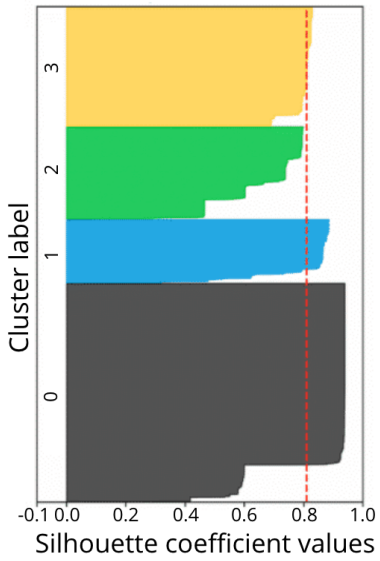

Fig. 12: Combined silhouette values which are distinguished by cluster.

VII. Citrus Evaluation

A. Ground Truth

This section evaluates the labelling approach used to derive a ground truth for LUFlow '20. As previously discussed, the approach taken performs clustering of graph-based features.

An example plot is presented in Figure 12, which shows silhouette values for various clusters of graph based features used to derive a ground truth. In this case, it is visible that the clusters are dense, well separated and consistent with each other. This is because the clusters are of similar thickness, indicating a similar sample size, and contain high silhouette

values which are all in the region of the silhouette average, indicated by the vertical red dashed line. This is further visualised in Figure 13, which shows the clusters more clearly by plotting the corresponding data points in feature space.

In order to fully evaluate how similar every supernode and outlier clusters are to each other, the average silhouette value is calculated for every graph used to label telemetry relating to LUFlow '20. As previously discussed, a graph is created for every date telemetry is captured to compile LUFlow '20. A time series of these values, ranging from June to October 2020, is presented in Figure 14, which displays an average silhouette value for variable cluster size, n.

As shown, the lowest average silhouette value is greater than 0.55, indicating clear cluster consistency in general. Figure 15 illustrates a box plot representing the distribution of silhouette values for varying number of clusters. Evidently, we can deduce that significant differences exist between the distribution of silhouette values based upon the size of the clusters chosen. The mean silhouette value for a cluster size of six is $\mu_6 = 0.7868$, and the standard deviation is $\sigma_6 = 0.0727$. When the cluster size is at the lowest, a value of two, the mean silhouette value is $\mu_2 = 0.7572$, and the standard deviation is $\sigma_2 = 0.0731$. Hence, it can be concluded that nodes within the supernode and outlier clusters truly belong in those clusters due to the absence of any negative silhouette metrics.

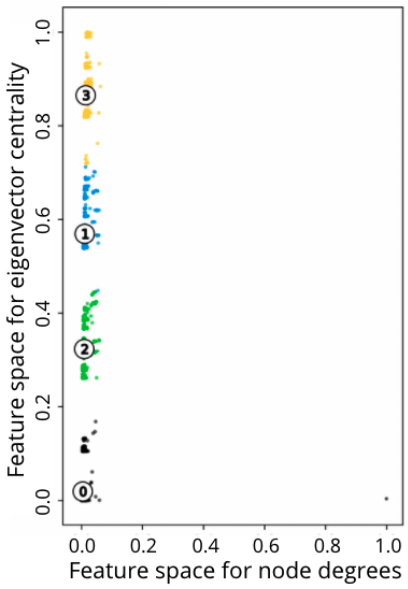

Fig. 13: Clusters plotted in feature space.

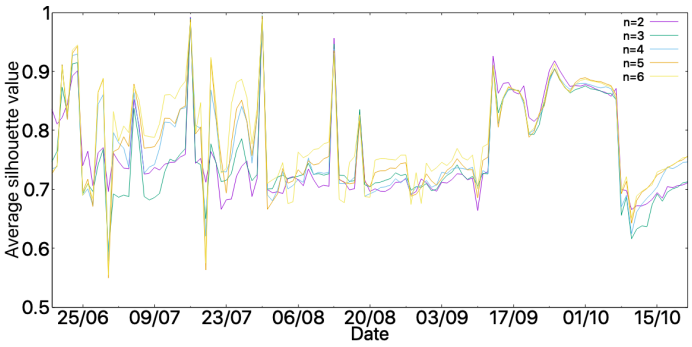

Fig. 14: Average silhouette value time series.

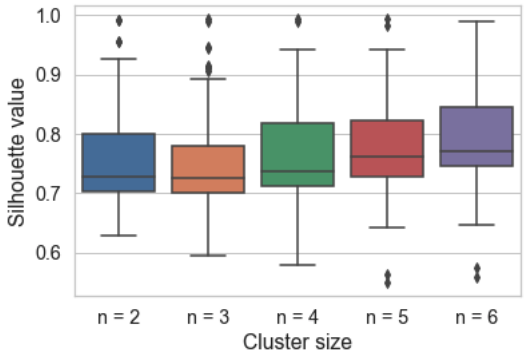

Fig. 15: Box plot depicting distribution of silhouette values for varying cluster sizes.

recall and F-score. Gradient boosting is a machine learning approach suitable for regression and classification problems which produces a model which is an ensemble of sequentially made weak prediction models. Due to the efficiency of the gradient boosted tree classifier to accurately detect both benign and malicious behaviour within LUFlow '20, it is chosen to perform online intrusion detection.

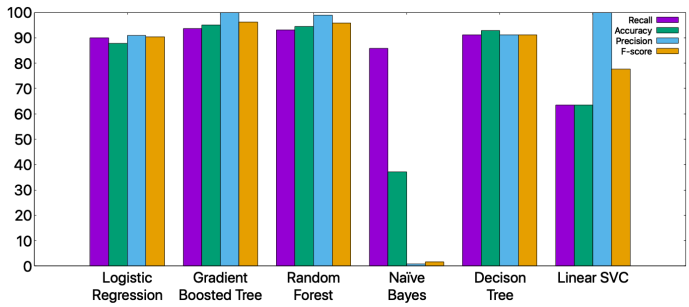

Fig. 16: Statistical metrics observed from offline detection.

B. Detection Performance

1) Offline Detection: The results of the offline detection approach is outlined in Figure 16. Each model listed within this figure has been trained using a grid search to identify the optimal parameters for this classification problem. As shown, the worst performing supervised classification algorithm was Naive Bayes. On the contrary, the algorithm which performed the best was the gradient boosted tree classifier, which obtained 99.98% precision, as well as high accuracy,

2) Online Detection: **Scanning**: As shown in Figure 17, these metrics showcase Clementine's high accuracy, 97.46%, and extremely high recall, 99.70%. This demonstrates the ability of Clementine to detect scanning attempts using live telemetry.

DDoS: The performance metrics associated with the classification is illustrated in Figure 17. As shown, Clementine performs extremely well under a DDoS attack; demonstrating a low number of FPs and FNs and over 99% in all calculated

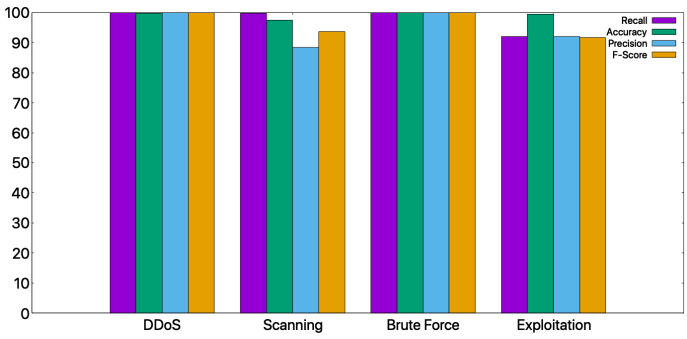

Fig. 17: Performance metrics obtained from online classification of live telemetry.

metrics.

Brute Force: As shown in Figure 17, Clementine is able to successfully distinguish between benign flows and flows relating to brute force attempts with extremely high precision, accuracy, and, recall.

Multi-stage Intrusion: As illustrated in Figure 17, Clementine classifies network flows under this attack scenario with very high accuracy, over 99%, with all other metrics achieving above 90%. Critically, flows associated with the Meterpreter reverse shell are correctly classified as malicious. As previously discussed, the malicious telemetry captured to create LUFlow '20 emanates from medium-interaction honeypots. Therefore, this specific attack does *not* exist in training data as payloads are captured but not executed, and can be treated as a novel attack vector.

C. Classification Performance

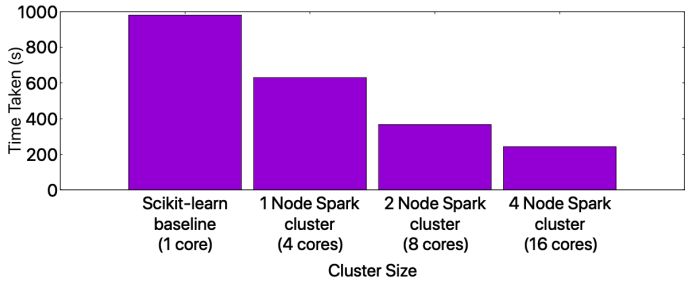

Fig. 18: Comparison of model selection using grid search and variable cluster size.

1) Model Training: As illustrated in Figure 18, scikit-learn's grid search implementation takes the longest time, 979 seconds, to find the optimal parameter combination. As previously discussed, this is because each model is trained sequentially. The parallel processing capabilities Spark provides are evident even on a single node cluster. Spark is able to leverage the 4 cores on a single machine, training models independently and reducing the total time taken compared to traditional approaches. This reduction in model training and selection is further evidenced in clusters of larger node sizes. Within a 4 node cluster, the average computation time is 243 seconds, a 75% reduction when compared to the baseline. This clearly demonstrates the performance benefits gained from leveraging Spark and a cluster of nodes to train and search for optimal model parameters.

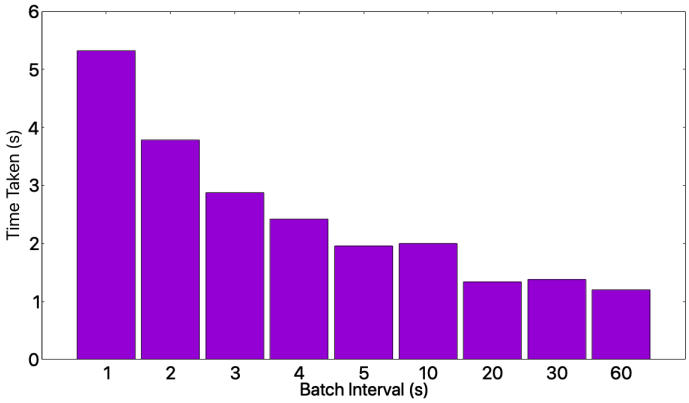

Fig. 19: Evaluation of classification efficiency using variable batch interval.

2) Streaming: As illustrated in Figure 19, the batch interval value has a significant impact upon the total processing time. In general, as the batch interval increases, the processing time decreases. For example, specifying a batch interval of 1 second separates the data into multiple mini-batches, all of which incur scheduling overheads. On average, the 1 second batch interval took $\mu_1 = 5.32$ seconds to classify all 100,000 records. In comparison, a batch interval of 60 seconds took an average of $\mu_{60} = 1.2$ seconds. This is because, in this case, all of the input data can be streamed within the batch interval, ensuring only a single batch to be processed. However, this interval means at least 60 seconds transpire before any classification occurs, and therefore does not allow near real-time results.

In order to further investigate the distribution of processing times, we perform additional analysis on these measurements. Figure 20 displays a Cumulative Distribution Function (CDF) plot for measurements taken with a batch interval of 2, 5, and 20 seconds. Notably, there exist differences between the distribution of the measurements with respect to the batch intervals. The mean processing time with a batch interval of 2 is $\mu_2 = 3.780$ seconds, and has a standard deviation of $\sigma_2 = 0.691$. When the batch interval is 5 seconds, the mean processing time is $\mu_5 = 1.96$, and has a standard deviation of $\sigma_5 = 0.268$. The mean processing time for an interval of 20 is $\mu_{20} = 1.34$, and has a standard deviation of $\sigma_{20} = 0.389$. Despite the large difference in interval times, it is apparent that intervals of 5 and 20 seconds exhibit similar statistical properties. In this instance, we recommend an interval of 5 seconds due to the lesser waiting time.

These results are extremely positive since they demonstrate Citrus' ability to rapidly process large amounts of data, and pave the way towards the composition of real-time detection. Due to this, we conclude Citrus can be leveraged to detect emerging threats in large-scale networked environments.

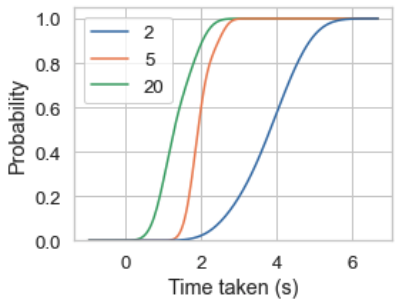

Fig. 20: Cumulative Distribution Function (CDF) for processing time distinguished by batch interval.

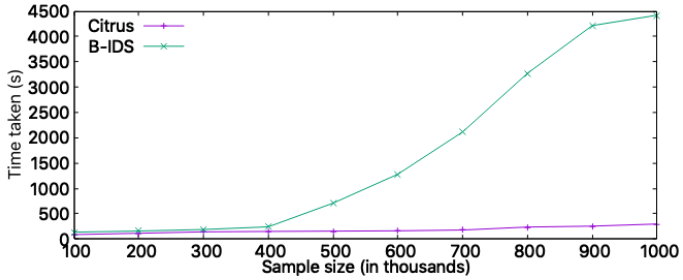

Fig. 21: Comparison of computational cost to label varying number of samples.

D. Experimental Comparison

Citrus establishes data ground truth through an unsupervised clustering approach. In contrast to supervised classifications methods which have been traditionally used in a multitude of systems (e.g., [13]), there is no requirement of costly training procedures. In order to demonstrate Citrus' advantages, an experimental comparison is performed between Citrus and a similar data set labelling framework, B-IDS [26].

We note that a single day of telemetry capture, comprised of one million records, is used in this evaluation to compare the approaches. We labelled this telemetry using the unsupervised and supervised approaches on the same hardware. The supervised B-IDS method was implemented using open-source software and was configured according to the specifications provided in [26].

Classification metrics were calculated for the online detection scenarios, outlined in Section VI-B, using the telemetry labelled by both Citrus and B-IDS. Evidently, the results show that Citrus outperforms the supervised alternative, and has an increased F-score metric by 1%. Due to F-score representing the harmonic mean between recall and precision, we conclude that Citrus' labelling methodology is superior in regards to enhancing detection mechanisms.

Furthermore, the experimentation has also demonstrated that Citrus has a more optimal computational cost when compared to the alternative. Figure 21 illustrates the time taken to label a number of samples using both approaches. As shown, there exists similar computational cost for the lower number of samples. However, when considering larger sample size, Citrus performs much better, indicating clear supremacy in this regard, and further supports its real-time detection pipeline.

VIII. Conclusion

This paper presents the design, implementation, and evaluation of a practical solution for real-time intrusion detection, Citrus. Citrus provides contributions to real-time detection through a novel unsupervised labelling method and a robust data pipeline which exploits properties of parallelism.

Citrus contains the Tangerine component which establishes data ground truth to automatically provide accurate target labels for the compiled data set, LUFlow '20. This data set is released to the general public through a GitHub repository in an attempt to aid further research efforts. An evaluation of the labelling approach is conducted to ensure the accuracy of the identified attack supernodes.

Furthermore, the Clementine component leverages this data set to provide practical intrusion detection capabilities using Spark's streaming abstraction. The work outlined in this paper showcases the accuracy of the detection capabilities, and demonstrates the efficiency of parallel computation through the evaluation and comparison of model training implementations.

References

[1] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets," vol. 1, no. 1, 2018. [Online]. Available: http://arxiv.org/abs/1806.03517

[2] J. Navarro, A. Deruyver, and P. Parrend, "A systematic survey on multi-step attack detection," *Computers & Security*, vol. 76, pp. 214–249, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818302141

[3] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and Big Heterogeneous Data: a Survey," *Journal of Big Data*, vol. 2, p. 3, 2015.

[4] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," *Proceedings of the 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2011*, pp. 29–36, 2011.

[5] K. Kendall and A. C. Smith, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems by A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," 1999.

[6] "Unified host and network dataset." [Online]. Available: https://csr.lanl.gov/data/2017.html

[7] R. Sommer and V. Paxson, "Outside the Closed World : On Using Machine Learning For Network Intrusion Detection," pp. 305–316, 2010.

[8] S. Abt and H. Baier, "Are We Missing Labels? A Study of the Availability of Ground-Truth in Network Security Research," *Proceedings - 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2014*, pp. 40–55, 2016.

[9] M. Baykara and R. Daş, "A Survey on Potential Applications of Honeypot Technology in Intrusion Detection Systems," *International Journal of Computer Networks and Applications (IJCNA*, vol. 2, no. 5.

[10] I. Yahya, M. Al, P. Chauhan, S. Shukla, and M. B. Potdar, "Review on efficient log analysis to evaluate multiple honeypots using ELK," no. 6, pp. 492–504, 2016.

[11] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, 2019.

[12] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[13] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2011.07.001

[14] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2015.

[15] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, 2019.
[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," 2018.
[17] M. Małowidzki, P. Berezi, and M. Mazur, "Network Intrusion Detection: Half a Kingdom for a Good Dataset," *Proceedings of NATO STO SAS-139 Workshop*, pp. 1–6, 2015.
[18] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Computers & Security*, vol. 70, pp. 238–254, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404817301165
[19] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Packet and Flow Based Network Intrusion Dataset," in *Contemporary Computing*, M. Parashar, D. Kaushik, O. F. Rana, R. Samtaney, Y. Yang, and A. Zomaya, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 322–334.
[20] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning Intrusion Detection: Supervised or Unsupervised?" in *Image Analysis and Processing – ICIAP 2005*, F. Roli and S. Vitulano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 50–57.
[21] J. Mchugh, "Testing Intrusion Detection Systems : A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," vol. 3, no. 4, pp. 262–294, 2001.
[22] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," no. Cisda, pp. 1–6, 2009.
[23] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab : Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking," Tech. Rep., 2010.
[24] A. Sperotto, R. Sadre, F. Van Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5843 LNCS, pp. 39–50, 2009.
[25] F. J. Aparicio-Navarro, K. G. Kyriakopoulos, and D. J. Parish, "Automatic dataset labelling and feature selection for intrusion detection systems," in *Proceedings - IEEE Military Communications Conference MILCOM*. Institute of Electrical and Electronics Engineers Inc., nov 2014, pp. 46–51.
[26] F. Gargiulo, C. Mazzariello, and C. Sansone, "Automatically building datasets of labeled IP traffic traces: A self-training approach," *Applied Soft Computing Journal*, vol. 12, no. 6, pp. 1640–1649, 2012.
[27] N. Nguyen and R. Caruana, "Classification with partial labels," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 551–559, 2008.
[28] W. Meng, Y. Liu, S. Zhang, D. Pei, H. Dong, L. Song, and X. Luo, "Device-agnostic log anomaly classification with partial labels," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–6.
[29] Y. Chen, X. Lian, D. Yu, S. Lv, S. Hao, and Y. Ma, "Exploring shodan from the perspective of industrial control systems," *IEEE Access*, vol. 8, pp. 75 359–75 369, 2020.
[30] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers and Security*, vol. 86, pp. 147–167, 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.06.005
[31] M. Zaman and C. H. Lung, "Evaluation of machine learning techniques for network intrusion detection," *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, pp. 1–5, 2018.
[32] J. Dromard and P. Owezarski, "Study and Evaluation of Unsupervised Algorithms Used in Network Anomaly Detection," *Advances in Intelligent Systems and Computing*, vol. 1070, pp. 397–416, 2020.
[33] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, M. A. Amanullah, I. Abaker Targio Hashem, E. Ahmed, and M. Imran, "Clustering-based real-time anomaly detection—A breakthrough in big data technologies," *Transactions on Emerging Telecommunications Technologies*, pp. 1–27, 2019.
[34] A. Marnerides, D. Prince, J. Couzins, R. Mills, V. Giotsas, P. McEvatt, D. Markham, and C. Irvine, "Fujitsu white paper: Cyber threat lab," 2019.
[35] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers and Security*, vol. 31, no. 3, pp. 357–374, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2011.12.012
[36] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers and Security*, vol. 45, pp. 100–123, 2014.
[37] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset Benchmark Dataset," no. July, 2017.
[38] J. Dromard and P. Owezarski, "Study and Evaluation of Unsupervised Algorithms used in Network Anomaly Detection," Tech. Rep. [Online]. Available: https://hal.laas.fr/hal-02334251
[39] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

Ryan Mills is a PhD Student within the School of Computing and Communications at Lancaster University. His research interests revolve around the profiling and detection of emerging threats. In order to provide innovative solutions to various challenges relating to these areas of research, he has investigated and leveraged highly scalable anomaly detection techniques.

Angelos K. Marnerides is Senior Lecturer (tenured Associate Professor) in the School of Computing Science at the University of Glasgow, UK and leading the Glasgow Cyber Defence Group (GCDG). His research revolves around applied and data-driven security and resilience for Internet-enabled cyber physical systems, the Internet at scale and programmable networks. His research has received significant funding from the industry (e.g., Fujitsu, BAE, Raytheon) and governmental bodies (e.g., EU, IUK, EPSRC). He has been a member of the IEEE and the ACM since 2007 and served as a Technical Program Committee (TPC) member, TPC track and workshop co-chair and organiser for several top-tier IEEE conferences (e.g., IEEE ICC, IEEE GLOBECOM) leading him to receive IEEE ComSoc contribution awards in 2016 and 2018. He obtained his PhD in Computer Science (2011) from Lancaster University and held lectureships, postdoctoral, and visiting researcher positions at Lancaster University (UK), Carnegie Mellon University (USA), University of Porto (Portugal), and University College London (UK).

Matthew Broadbent Matthew Broadbent's research interests are broadly focused around the intersections of computer networking, security and multimedia. The research he has conducted in these areas has focused on the application of software-defined networks and flexible infrastructures; an emerging research topic that has garnered much attention from both academia and industry.

Nicholas Race is Professor of Networked Systems within the School of Computing & Communications at Lancaster University. He is also the Director of the Cyber Security Research Centre, and the Associate Dean for Research in the Faculty of Science Technology at Lancaster. His research focuses on developing future networking services built upon Software Defined Networks and Network Functions Virtualisation. This includes new techniques to enhance the Quality of Experience (QoE) of media streaming and support for the detection and remediation of network anomalies. He has a particular interest in the use of these concepts for monitoring and security, building upon his previous work in developing lightweight intrusion detection mechanisms and security monitoring for Wireless Mesh Networks. He is Principal Investigator of NG-CDI, a £5M EPSRC Prosperity Partnership research programme with BT that aims to develop a future network that is "autonomic", with the capability to react and reconfigure infrastructure accordingly with minimal human intervention. Previously he has played leading roles on many other large-scale UK and European projects (including TOUCAN, INITIATE, MPAT, FI-CONTENT2, STEER, Fed4FIRE, P2P-Next and OFELIA).