Distributed, Multi-Level Network Anomaly Detection for Datacentre Networks

Mircea Iordache*, Simon Jouet*, Angelos K. Marnerides† and Dimitrios P. Pezaros*
*School of Computing Science, University of Glasgow, G12 8QQ, UK
†InfoLab21, School of Computing & Communications, Lancaster University, LA1 4WA, UK
Email: m.iordache-sica.1@research.gla.ac.uk, angelos.marnerides@lancaster.ac.uk,
{simon.jouet, dimitrios.pezaros}@glasgow.ac.uk

***Abstract*—Over the past decade, numerous systems have been proposed to detect and subsequently prevent or mitigate security vulnerabilities. However, many existing intrusion or anomaly detection solutions are limited to a subset of the traffic due to scalability issues, hence failing to operate at line-rate on large, high-speed datacentre networks. In this paper, we present a two-level solution for anomaly detection leveraging independent execution and message passing semantics. We employ these constructs within a network-wide distributed anomaly detection framework that allows for greater detection accuracy and bandwidth cost saving through attack path reconstruction.Experimental results using real operational traffic traces and known network attacks generated through the Pytbull IDS evaluation framework, show that our approach is capable of detecting anomalies in a timely manner while allowing reconstruction of the attack path, hence further enabling the composition of advanced mitigation strategies. The resulting system shows high detection accuracy when compared to similar techniques, at least 20% better at detecting anomalies, and enables full path reconstruction even at small-to-moderate attack traffic intensities (as a fraction of the total traffic), saving up to 75% of bandwidth due to early attack detection.**

I. Introduction

In recent years, unwarranted network accesses have raised serious concerns for the Internet. Distributed Denial of Service (DDoS) attacks pose a great risk to network security, but with the increased exposure of implementation errors in critical applications, other attacks are starting to become ever more prevalent. These new attack vectors are difficult to detect early and the time to resolve the vulnerabilities is uncertain. In many environments, patches for these exploits require approval from service and package maintainers, or interfere with operation critical components, increasing the time until the issue is fully mitigated. In order to prevent such attacks from occurring, it is crucial to properly detect them from within a large volume of aggregate data. The detection is complex due to the compute and network requirement to isolate anomalous from legitimate traffic without degrading the network performance. The problem is further amplified by the nature of general network anomalies which are increasingly difficult to characterise through a generalised pattern due to their continuously evolving nature [1]. There has been particular proliferation of DDoS attacks in recent years since they can be easily performed, require little technical know-how, can hinder the infrastructure for a long time, and are hard to detect and mitigate [2]. This type of attack is characterised by a large number of flows from many-hosts to a destination server and service, much like a legitimate flash-crowd event. Alternatively, another type of attack such as a port-scan, while not interfering with network operations, can be used for malicious purposes. Such an attack is characterised by a large number of short lived unidirectional flows over a large range of ports. IDS have been proposed to detect these attacks but often have to trade scalability for accuracy and vice versa. Fine-grained network monitoring yields better detection results but can degrade performance and prevent scalability, while coarse-grained monitoring can be used over large amount of data with lower confidence on detecting anomalies. Over the last decade, with the introduction of Software Defined Networking (SDN), the network management, configuration, and monitoring have been greatly improved, enabling new ways to tackle anomaly detection.

In this paper, we propose a two-level approach for network anomaly detection [2], [3], which combines coarse-grained monitoring performance with the accuracy of fine-grained analysis, resulting in improved detection accuracy with minimal performance costs. We further enhance the architecture by enabling execution within a distributed environment. Our motivation is supported by the fact that network anomalies present two overarching characteristics, albeit difficult to parameterise: flows with deviations from typical network traffic, and asymmetry of interaction between end points. Although several solutions have been proposed for network anomaly detection, we propose an enhancement of the two-level detection scheme through use of distributed computing. Thus, our approach differs from others in the following ways:

a) *Accuracy:* While other statistical anomaly detection mechanisms provide suitable accuracy, our approach, through using a two-level detection and leveraging distributed mechanisms, gives a higher degree of accuracy.
b) *Path Reconstruction:* Using modular distributed components for anomaly detection, our solution is able to perform path reconstruction.
c) *Memory Consumption:* We provide an adequate trade-off between memory consumption and performance by using data structures that attempt to provide an accurate representation of flows for indexing and statistical manipulation.

The remainder of this paper is structured as follows: In Section II, we describe the design and implementation of the proposed anomaly detection algorithms and distributed mechanisms. Section III presents the experimental setup and results obtained, with emphasis on scalability, accuracy, and path reconstruction. Section IV discusses related work in the

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

field of anomaly detection, and Section V concludes the paper.

II. Design & Implementation

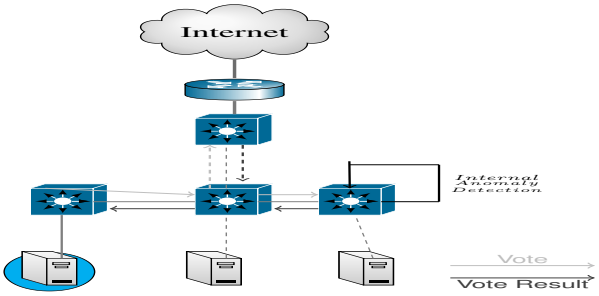

Fig. 1. A high-level design of the proposed system architecture

As shown in figure 1, the proposed anomaly detection solution is structured around the use of different modules responsible for detecting diverse anomalies but with different performance and computational characteristics. They are distributed across the infrastructure to have a global view of the network. Ideally, each module would be deployed at every network node, either directly on or collocated with the switch platform to analyse data at every layer and at different levels of aggregation. This is reminiscent of legacy network firewall boxes, where processing was done as close to the networking layer as possible. This approach, however, lacked flexibility in deployment to multiple network locations, which we improve through software able to run on general purpouse hardware. Furthermore, we use this approach to maximise data locality and ensure minimum networking overhead and redirection when analysing traffic.

In order to improve detection accuracy without hindering performance, we rely on a two-level approach at anomaly detection: each module independently computes certain traffic statistics and it exchanges meta-information with its peers through a Remote Procedure Call interface we have developed. Monitoring peers subsequently run a consensus algorithm to collectively identify whether the traffic is anomalous or not. Using RPC, we can distribute the network processing across many devices increasing the computational capabilities as well as providing path reconstruction of the anomaly source by analysing the sequence of RPC control messages and the voting responses from the consensus algorithm.

A. Fine and Coarse-Grained Anomaly Detection

The proposed anomaly detection method is following a two-level detection scheme. Using this approach, we can have a trade-off between anomaly detection accuracy and computational cost. First, a coarse-grain inspection of the traffic is performed for fast-analysis, minimising processing overhead for benign traffic. If a potential anomaly is detected, a fine-grained detection algorithm is used to analyse the collected metrics of the potentially anomalous subset of traffic in-depth. In order to enable high performance for a wide variety of environments, the two algorithms are independent of network topology and the underlying infrastructure.

For Coarse-Grained Detection (CGD), we employ Shannon's statistical entropy-based method which is independent of network topology and traffic characteristics, and can be used for anomaly detection and classification purposes [4]. The Shannon entropy $H(X)$ of a data set $X = x_1, x_2, ...x_n$ is defined as

$$H(X) = -\sum_{i=1}^{N} p_i log_2(p_i)$$

where $N = |X|$ and p_i is the probability $P[X = x_i]$; it is used to measure the expected values of information conveyed through data set. Further improvements for use of anomaly detection are defined through normalisation of values, thus

$$H'(X) = -\sum_{i=1}^{N} \frac{p_i log_2(p_i)}{log_2 N}$$

ensures that the entropy values are in the range $[0, log_2 N]$. Statistical anomaly detection algorithms usually compose a normal behaviour model in order to establish the ground truth with respect to the normal behaviour of a given network by profiling a set of network features over a long period of time. Hence, with the use of the normal behaviour model, it is then feasible to identify anomalous patterns if the examined network features deviate from certain thresholds of the expected values. There is usually a great debate on the amount of features to be used within a network anomaly detection methodology, however, such features are usually extracted from the network or the transport layer (layer 3–4) [1]. In our proposed design, we focus on the traditional 5-tuple flow definition containing the protocol type as well as source and destination addresses and ports. Hence, in our case any entropy distribution change outside of a set threshold is an indicator of a possible anomaly. The accuracy of the detection can be improved by analysing the variation of multiple iterations of Shannon's entropy. Once the entropy variation has passed the thresholds of normal behaviour which is dependant on network load and characteristics, a more detailed inspection of the network traffic can be performed to isolate the true- and false-positives as well as to identify the type of anomaly.

For in-depth analysis, we have selected a sketch-based algorithm for fine-grained detection. For this work, we modify the Count-Min Sketch [5] algorithm to store the source and destination IP addresses, as well as the transport layer ports. Under this approach, we can store the time variant state of the flow. Any variation present within the flows can be detected from the stored metrics and used to detect the presence of an anomaly. As a result, overall network anomalies can be detected as well as per-flow-specific anomalies, with the added benefit that we are able to use the resulting information for anomaly classification purposes. Using this approach to complement Shannon's entropy estimation, we are able to accurately identify anomalies and gather detailed network characteristics necessary to ensure consensus on the detected anomaly being reached.

B. Flow Statistics Collection

The Flow Statistics Collection module is responsible for receiving flow information from the network controller. Since the anomaly detection modules rely on the 5-tuple, the collector presents metrics from the network and transport layers, as well as associated metrics representing the state of each flow.

The additional metrics we are using are the per-flow number of packets and bytes transmitted since they give an overall accurate indication of network utilisation and anomalous flows.

Using the OpenFlow protocol as our underlying network orchestration mechanism, we configure the switches to hold flow entries at layer 4, effectively mapping each flow entry to the flow 5-tuple. Using the OpenFlow **FlowStats** query message, the system is able to gather per flow entry metrics including packet and byte counts.

The implementation of the orchestration mechanisms used to manage the network components is done in the form of an OpenFlow Controller application, instructing switches to gather detailed information of the aforementioned metrics.

C. Distributed Behaviour

While the proposed anomaly detection components described so far can be used in a standalone fashion, we significantly increase the performance and efficiency of our solution through including distributed behaviour. This enhances the overall system by allowing attack path reconstruction and coordinated analysis from multiple network locations. In order to describe certain aspects of the mechanisms used, we need to establish certain prerequisites and assumptions about the distributed behaviour we compose:

- *Independence:* Anomaly detection instances should perform independently from other identical processes throughout the overall system. This allows for reliability in results and minimisation of faulty processes within the distributed environment with respect to data collection, analysis, and response.
- *Ordering:* In order to ensure correct behaviour, establishing and maintaining an accurate ordering of operations performed is critical. This enables path reconstruction of detected anomalies and victim pinpointing through reasoning behind the ordering of events and actions performed by the distributed system.
- *Consensus:* To enable precise detection of attacks, agreement between members of the distributed environment must be reached. In the case of our system, this is achieved through the use of a majority voting scheme that gathers the results from all instances involved in the detection process.

The main component that enables distributed behaviour is the use of a remote procedure call mechanism allowing for analysis of network data from different devices. Execution of our system in a distributed environment enhances detection accuracy, and allows for monitoring of events to perform attack path reconstruction, facilitating the use of improved mitigation strategies that minimise network bandwidth usage.

In order to provide a clear explanation regarding remote invocation, we have to impose some constraints on the behaviour of our solution and a discussion on locality. The behaviour is dictated by events present within the network, which in the case of the studied problem are network anomalies that have been detected. Using the Communicating Sequential Processes (CSP) notation, we define the behaviour of our solution as a series of processes that all implement the same functionality, and rely on shared events that model inter-process communication. The formal definition for our system under the stated constraints is given by Theorem 1, with proof provided in Proof 1. The event $anom$ is external to the distributed system, while events $analyse$ and $vote$ convey messages that are shared between multiple instances of the prototype we have developed, abstracting remote invocation semantics.

Theorem 1: Given two processes:

$$Background = (anom \rightarrow analyse)\Box(vote \rightarrow STOP)$$

$$Conditional = analyse \rightarrow vote \rightarrow STOP$$

a series of events $anom$, $analyse$ and $vote$, and a strict ordering $anom \rightarrow analyse$ and $analyse \rightarrow vote$, a system comprised of these processes will behave deterministically.

$$Background|\{analyse\}|Conditional|\{vote\}|Background \equiv anom \rightarrow analyse \rightarrow vote \rightarrow STOP$$

Proof 1: Using the operational semantics rules for alphabetised parallel.

We appeal to the operational semantics rules of alphabetised parallel: two processes P and Q with respective alphabets A_P and A_Q and $x \in A_P \cap A_Q, y \in A_P \setminus A_Q$ will behave as follows: if $P \rightarrow^y P'$ then $P|\{x\}|Q \rightarrow^y P'|\{x\}|Q$ and if $P \rightarrow^x P'$ and $Q \rightarrow^x Q'$, then $P|\{x\}|Q \rightarrow^x P'|\{x\}|Q'$. Thus:

$$\begin{aligned}
&Background|\{analyse\}|Conditional \\
&\equiv (anom \rightarrow analyse) \\
&\Box(vote \rightarrow STOP)|\{analyse\}|(analyse \rightarrow vote \rightarrow STOP) \\
&\equiv (anom \rightarrow analyse)|\{analyse\}|(analyse \rightarrow vote \rightarrow STOP) \\
&\Box(vote \rightarrow STOP)|\{analyse\}|(analyse \rightarrow vote \rightarrow STOP) \\
&\equiv (anom \rightarrow analyse \rightarrow vote)\Box(vote \rightarrow STOP).
\end{aligned}$$

The second operand of $\Box$ implies no synchronization between the two processes, and is therefore equivalent to $vote \rightarrow STOP$. Similarly:

$$\begin{aligned}
&Conditional|\{vote\}|Background \\
&\equiv (analyse \rightarrow vote \rightarrow STOP)\Box(anom \rightarrow analyse).
\end{aligned}$$

By abstracting events, and imposing ordering such that we enforce $anom \rightarrow analyse$ and $analyse \rightarrow vote$, we can define an improved behaviour of our system:

$$\begin{aligned}
&Background|\{analyse\}|Conditional \setminus \{vote\} \\
&\equiv anom \rightarrow analyse \rightarrow vote
\end{aligned}$$

$$\begin{aligned}
&Conditional|\{vote\}|Background \setminus \{analyse\} \\
&\equiv analyse \rightarrow vote \rightarrow STOP
\end{aligned}$$

Finally, by combining the two we can deduce the following order, which can be described as deterministic:

$$\begin{aligned}
&Background|\{analyse\}|Conditional|\{vote\}|Background \\
&\equiv anom \rightarrow analyse \rightarrow vote \rightarrow STOP
\end{aligned}$$

Although the formalised description on behaviour does not describe the difference between the two processes, we can control them through an analysis of network topology. In our

solution, we define the $Background$ process as operating on network end-points, where hosts are directly connected to switches, and $Conditional$ as locations which are connected only to other switches and routing devices. Thus, by introducing constraints derived from the network topology, we are able to predict that detection on non-end-point devices which are running $Conditional$ processes, only occurs if the anomaly has been detected by a $Background$ instance. Furthermore, if we expand this by requiring that a process can only communicate to another only if they are adjacent in the network topology, we can easily deduct the path reconstruction capabilities of our solution, by allowing a $Conditional$ process to invoke one another through $analyse$ events.

Given the ordered nature of the messages passed they can be traced back to a detection single instance, as per the construction we use in Theorem 1. The use of a decentralised majority voting solution does not yield any benefits; because of this, our approach uses a centralised version of the proposed algorithm, with only a single voting phase, as described in Algorithm 1. This centralised voting process occurs within the context of all nodes which have received an $analyse$ event during the occurrence of an anomaly, which we define as a group; the central node within a group is the one that initiated the invocation, thus, becoming the last one to receive a $vote$ event.

Algorithm 1 Single-phase Majority Voting. Notations described in Table I

```
Given G(a_i) and C(a_i)
if |C(a_i)| > |G(a_i)|/2 then
    S_i ← M
else
    S_i ← F
end if
```

a_i	central node in a group
S_i	state of a_i, $S_i \in \{M, F\}$ M=Majority, F=Fewness,
$M(a_i)$	set of group members of a_i being labelled with M
$F(a_i)$	set of group members of a_i being labelled with F
$G(a_i)$	set of group members of a_i, where $G(a_i) = M(a_i) \cup F(a_i)$ and $\|G(a_i)\| = n$
$C(a_i)$	set of group members of a_i belonging to the same cluster (voted the same) as a_i

TABLE I. NOTATIONS FOR THE VOTING METHOD.

Our prototype, including the IDS, Controller module (developed for the Ryu OpenFlow Controller), and afferent utility scripts and applications can be found on GitHub[1]

III. EVALUATION

A. Experimental Setup

Our performance evaluation experiments were based on CAIDA's 2014 Anonymised Internet Traces [6], while the attack traffic was obtained using Pytbull, a popular IDS testing software which uses several attack vectors. Regarding the background traffic used, we split the provided data through Monte-Carlo sampling to obtain realistic network traces more suitable for a datacentre network environment, as the CAIDA traces contain Internet traffic which encompasses communication between multiple sub-networks and does not account for inter-network communication. Furthermore, by combining traffic from both directions for the dataset used, we are able to realistically model intra-network traffic.

Our testbed uses an Intel Core i5-4670k 3.4GHz CPU with 16GB RAM running the Mininet Virtual Machine with 12GB RAM dedicated to the VM. We chose the Ryu Openflow controller, mainly due to the simplicity of development and flexibility regarding deployment and management. In Figure 2, we present the general network topology used for anomaly detection. Each of the switches is running our prototype, with end-point Switches 1-4 continually analysing gathered statistics for potential anomalies, while Switches 5-7 are only involved in the detection process if specifically invoked.

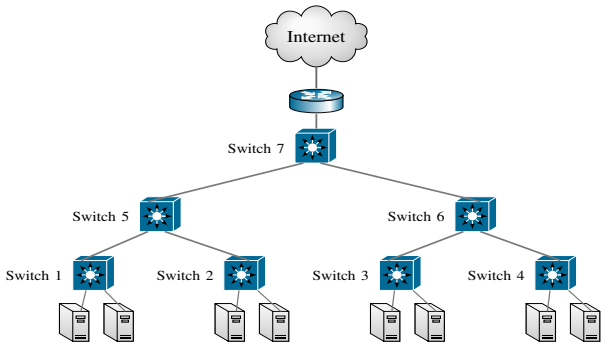

Fig. 2. Tree topology for complex testing and modelling interactions.

From the large range of possible attacks that can be evaluated, we decided to use a subset of available anomalies that represent a wide variety of attacks and allow us to accurately measure the performance of our proposed solution: Brute Force access which contain comparable patterns to 0-day attacks, DDoS attacks which are already being thoroughly investigated in other approaches [3], and Port Scans. This wide spread set of vulnerabilities contain features with large variations between them, therefore allowing us to thoroughly test the applicability of our prototype to the different styles of anomalies.

B. Detection Accuracy

To properly assess the detection accuracy of the selected algorithms, in Figure 3 we plot the Receiver Operating Characteristic curves, in terms of true- and false- positive rates produced by coarse-grained (entropy-based) and fine-grained detection (Sketch-based). In order to generate this curve, we benchmarked each implementation to assess the number of true and false positive alerts produced during multiple attacks with varying thresholds. Thus, we can confirm a high degree of accuracy in the detection capabilities of our proposed solution with minimal false positive events, also allowing for mitigation strategies that do not impact the services of the underlying network.

We also evaluated our system's performance with respect to false negative rates, which the Receiver Operating Characteristic does not take into account under realistic conditions. In order to better understand the performance of our system, we measure detection accuracy against anomaly intensity. In order

[1] https://github.com/mirceaIordache/DistributedNAD

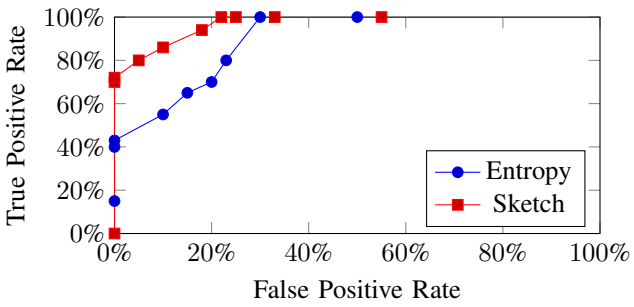

Fig. 3. Receiver Operating Characteristic for the entropy-based and Sketch-based anomaly detection algorithms.

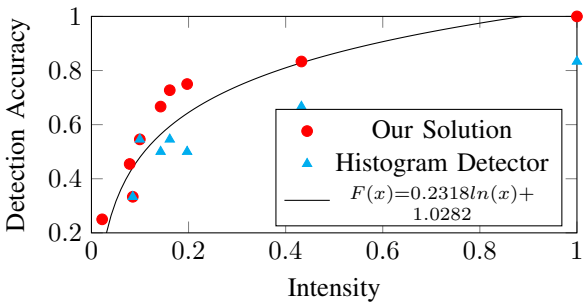

Fig. 4. Detection accuracy against attack intensity for the anomaly detection algorithms.

to do so, we define the anomaly intensity $I(\Delta t) = \frac{T_A(\Delta t)}{T_N(\Delta t)}$, where $T_A(\Delta t)$ is the anomalous traffic that is present during time interval $[0, \Delta t)$, and $T_N(\Delta t)$ is the overall amount of traffic flowing through a switch during the interval $[0, \Delta t)$. Similarly, we define the detection accuracy for this case as $A(\Delta t) = \frac{D_A(\Delta t)}{N_A(\Delta t)}$, where $D_A(\Delta t)$ is the number of detected anomalies in the time interval $[0, \Delta t)$, and $N_A(\Delta t)$ is the total number of anomalies occurring during the interval $[0, \Delta t]$.

Time (Δt)	Total Traffic (T_N)	Anomalous Traffic (T_A)	Total Intensity (I)	Total Anomalies (N_A)	Detected Anomalies (D_A)	Detection Accuracy (A)
12	1934	1934	1	6	6	1.00
12	4472	1934	0.432469	6	5	0.83
12	13576	1934	0.142457	6	4	0.66
24	1246	1246	1	4	4	1.00
24	6322	1246	0.19709	4	3	0.75
24	56445	1246	0.022075	4	1	0.25
30	140452	22654	0.161294	11	8	0.72
48	227434	22654	0.099607	11	6	0.54
56	287706	22654	0.07874	11	5	0.45
52	98027	8344	0.085119	3	1	0.33
60	98027	8344	0.085119	3	1	0.33
66	98027	8344	0.085119	3	0	0.00

TABLE II. EXPERIMENTAL SETUP AND OBSERVATIONS REGARDING ATTACK INTENSITY AND DETECTION ACCURACY.

In Table II, we have presented our observations obtained during several experimental sessions that contained multiple attacks and varying levels of background traffic. The abundance of low intensity attacks in our data is representative of actual network conditions, where a port scan or brute force attempt would consume little bandwidth while still acting as an anomalous event within the network, while attacks of higher intensity are akin to DDoS attacks where network bandwidth usage is significant.

In Figure 4, we also plot the findings obtained through comparison with a Histogram-based detection scheme which uses the Kullback-Leibler distance [7]. In order to maintain real-time online functionality, frequent detection intervals are required. These are, however, approximately two orders of magnitude higher than those presented in the original histogram implementation, which leads to higher standard deviation $\hat{\sigma}$ and thresholds, thus leading to lower overall detection accuracy.

C. Impact of Distributed Behaviour

Although we did not find statistically significant evidence of different detection capabilities when utilising distributed behaviour or a centralised instance, as the anomalous data analysed by multiple instances often coincides, the former approach is the main component that enables path reconstruction. When analysing the number of invoked instances for anomaly detection, we have come to the conclusion that, given an attack of adequate intensity, we are able to provide full path reconstruction as evidenced by data shown in Figure 5. In the case of a lower intensity attack, our solution is still able to provide partial path reconstruction capabilities, diminishing as increased volumes of legitimate traffic are making detection of anomalies increasingly difficult. With moderate intensity, our approach is capable of accurately determining the attack path, thus allowing for highly localised mitigation to occur. Based on this data, our solution is able to offer complete path reconstruction at the onset of DDoS attacks which generally have high intensity, and partial path reconstruction for anomalies of lower intensity. This functionality is enabled through analysis of the instances involved in the detection process. By providing mechanisms that allow mitigation of an attack close to its origin, our solution allows for enhanced bandwidth usage for legitimate activities.

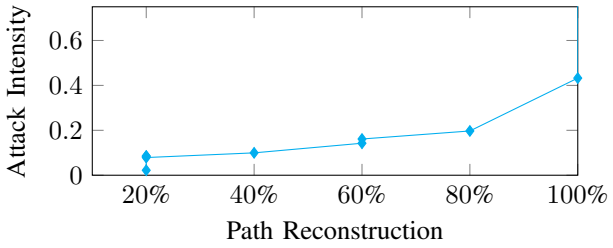

Fig. 5. Path Reconstruction capabilities based on attack intensity.

In terms of efficient mitigation, knowing the features of the anomalous flows [7] does not necessarily imply availability of such approaches. To achieve such results, given a global overview of routing tables as found within Software Defined Networks, requires advanced processing of labeled multigraphs in a fashion similar to network restoration, thus incurring additional delay between anomaly detection and mitigation and decreasing overall service capabilities. Our approach does not incur these overheads due to path reconstruction, allowing for timely mitigation. In our experiments, the delay between anomaly detection and path reconstruction was approximately 4 seconds.

Per-link bandwidth consumed by anomalous traffic during

our experiments was up to 35% (35Mbps out of 100Mbps) at the core layer and up to 75% at lower layers, as can be seen from Figure 6. Existing solutions filter detected anomalies at every network ingress, an approach which is susceptible to intra-network attacks in the case of large commercial datacentres. The other technique commonly used is filtering at the affected host or subregion of the network, which leads to wasted bandwitdth for legitimate activities, up to 65% of available bandwidth for the targeted host. Through path reconstruction, we can pinpoint the origin location of the anomaly, thus enabling the use of mitigation strategies that maximise the availability of the overall network, thus increasing performance.

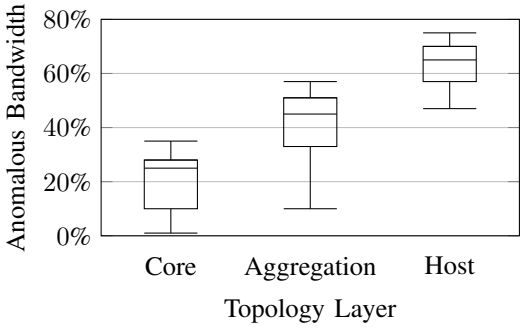

Fig. 6. Attack bandwidth at different network layers.

IV. Related Work

There is a large body of work focusing on network intrusion detection with multiple approaches, all using different types of information and processing ranging from statistical analysis, to Bayesian networks, Markov chains, and neural networks [8]. Many recent solutions for anomaly detection use sketches, as evidenced in [9] and [10]. A Sketch [11] is a data structure used to store a summary of large data in a memory efficient way. Cormode et al. [5] provided an approach for using multiple hash tables for scalable data stream summaries, proving a simplified version of the initial sketch.

Since our solution factors-in distributed behaviour, reaching consensus between the multiple components involved requires a majority voting scheme. Previous work on this topic is extensive, with diverse applications having specialised solutions [12]. In our work, we initially reviewed a decentralised two-phase majority voting scheme as described in [13], which analyses performance anomalies in networked environments. Based on data collected through different monitoring mechanisms, the participants use a two-phase majority voting scheme to decide on the state of all nodes within the group.

V. Conclusions

In this paper, we have investigated distributed network anomaly detection as a means to enhance network security in a scalable manner. Specifically, we exploited a two-level detection scheme to minimise the search space and time required for analysis of network data. For anomaly detection, we adopted an entropy-based approach combined with a Sketch data structure and algorithm for detailed analysis in order to improve the accuracy of identified attacks. We have proposed a distributed architecture that enables path reconstruction capabilities in a reliable way, and allows for higher degree of confidence in detected results through a majority voting scheme. The performed evaluation shows that our solution is resistant to low-intensity attacks and is capable of path reconstruction by maintaining the ordering of passed messages.

Future work will focus on enhancing the synchronisation process of remotely invoked instances through an advanced majority voting scheme that allows for extended information that includes the previously mentioned classification process. Finally, we wish to investigate enhanced deployment and management techniques for our solution to take into account data locality, system resource availability, and demand, guaranteeing enhanced resilience of overall network infrastructures.

Acknowledgments

The work has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/L026015/1, EP/N033957/1, EP/P004024/1, and EP/L005255/1, and by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS – Resilient communication services protecting end-user applications from disaster-based failures.

References

[1] A. Marnerides, A. Schaeffer-Filho, and A. Mauthe, "Traffic anomaly diagnosis in internet backbone networks: A survey," *Computer Networks*, vol. 73, pp. 224 – 243, 2014.

[2] H. Liu, Y. Sun, and M. S. Kim, "A scalable ddos detection framework with victim pinpoint capability," *Journal of Communications*, vol. 6, no. 9, pp. 660–670, 2011.

[3] K. Giotis, G. Androulidakis, and V. Maglaris, "A scalable anomaly detection and mitigation architecture for legacy networks via an openflow middlebox," *Security and Communication Networks*, 2015.

[4] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *Network, IEEE*, vol. 23, no. 1, pp. 6–12, 2009.

[5] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[6] U. CAIDA, "Anonymized internet traces 2014 dataset," 2014.

[7] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 6, pp. 1788–1799, 2012.

[8] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[9] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 187–200.

[10] R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. Dinda, M.-Y. Kao, and G. Memik, "Reverse hashing for high-speed network monitoring: Algorithms, evaluation, and applications," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. IEEE, 2006, pp. 1–12.

[11] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Quicksand: Quick summary and analysis of network data," Technical Report, Dec. 2001. citeseer. nj. nec. com/gilbert01quicksand. html, Tech. Rep., 2001.

[12] M. J. Fischer, "The consensus problem in unreliable distributed systems (a brief survey)," in *Foundations of Computation Theory*. Springer, 1983, pp. 127–140.

[13] L. Yu and Z. Lan, "A scalable, non-parametric method for detecting performance anomaly in large scale computing."