Anomaly Detection for Science DMZs Using System Performance Data

Ross Gegan*, Christina Mao*, Dipak Ghosal*, Matt Bishop*, Sean Peisert *†
* University of California, Davis
{rkgegan, cmao, dghosal, mabishop, speisert}@ucdavis.edu
†Lawrence Berkeley National Lab
{sppeisert}@lbl.gov

***Abstract*—Science DMZs are specialized networks that enable large-scale distributed scientific research, providing efficient and guaranteed performance while transferring large amounts of data at high rates. The high-speed performance of a Science DMZ is made viable via data transfer nodes (DTNs), therefore they are a critical point of failure. DTNs are usually monitored with network intrusion detection systems (NIDS). However, NIDS do not consider system performance data, such as network I/O interrupts and context switches, which can also be useful in revealing anomalous system performance potentially arising due to external network based attacks or insider attacks. In this paper, we demonstrate how system performance metrics can be applied towards securing a DTN in a Science DMZ network. Specifically, we evaluate the effectiveness of system performance data in detecting TCP-SYN flood attacks on a DTN using DBSCAN (a density-based clustering algorithm) for anomaly detection. Our results demonstrate that system interrupts and context switches can be used to successfully detect TCP-SYN floods, suggesting that system performance data could be effective in detecting a variety of attacks not easily detected through network monitoring alone.**

***Index Terms*—Science DMZ, data transfer node, high-performance computing, system performance metrics, anomaly detection, DoS attack, computer security, machine learning, scientific workflows, DBSCAN, clustering.**

I. Introduction

Modern research often requires the efficient and reliable movement of vast amounts of data, with some organizations generating terabytes of data daily [1]. To help facilitate the movement of petabytes of data, organizations utilize Science Demilitarized Zone (DMZ) networks - a specialized network model intended to maximize data transfer efficiency. Through a combination of network organization, performance tuning, and dedicated data transfer nodes (DTNs), the Science DMZ model helps guarantee reliable and high performance data transfers [1]. This also implies that protecting the performance of such networks is an important security concern, and security measures must be considered with this in mind [2]. Therefore, Science DMZs avoid typical firewalls to maximize network transfer efficiency, instead relying on various detection systems and Access Control Lists (ACLs) [2]. Typically, network intrusion detection systems (NIDS), such as Zeek(Bro) [3] or Snort [4], tend to rely solely on network metrics to identify abnormal traffic or attacks. However, we believe system performance metrics can also reveal the type of traffic being received, including malicious traffic [5] [6].

DTNs can become a critical point of failure in a Science DMZ if performance becomes compromised due to a denial-of-service (DoS) attack. Our work evaluates how system performance metrics might be used to identify a standard DoS attack such as a TCP-SYN flood attack, directed toward a DTN. In our study, we configured a server with a 10 Gbps backbone link to our campus Science DMZ as a DTN following the best practices outlined in [7]. Scientific workloads are emulated on the DTN by transferring files from the Energy Sciences Network's (ESnet) test DTNs, to generate real network traffic and system activity, while logging system performance metrics such as interrupts, CPU utilization, memory utilization, context switches as well as packets received/transferred and bytes transferred/received. We used an anomaly detection system based on DBSCAN clustering [8] to analyze these metrics for anomalies. The results given by the detection system are then evaluated to gauge the effectiveness of these metrics in detecting TCP-SYN floods as anomalies. By validating this detection method using a well-known and heavily studied attack, we establish the effectiveness of this method and that it can be extended upon and generalized to detect other types of external network-based attacks and insider attacks.

II. Science DMZ

A Science Demilitarized Zone (DMZ) is a network paradigm designed for enabling large-scale scientific research by providing a scalable environment for data transfers and reliable performance [2]. It has been recognized by the National Science Foundation (NSF) as a best practice and adopted at a number of research institutions in the United States to facilitate data-intensive scientific research [9]. Although Science DMZs vary among organizations depending on their purpose [10], there are some shared features across all DMZs. The Science DMZ is usually placed at the organization's network perimeter, with a border router connecting the site with the Science DMZ resources and typically has dedicated network

This material is based upon work supported by the National Science Foundation under Grant Number OAC-1739025 and by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors of this work

components tuned to maximize data transfer performance. Figure 1 shows a basic science DMZ setup with its various components. The most critical component of a Science DMZ

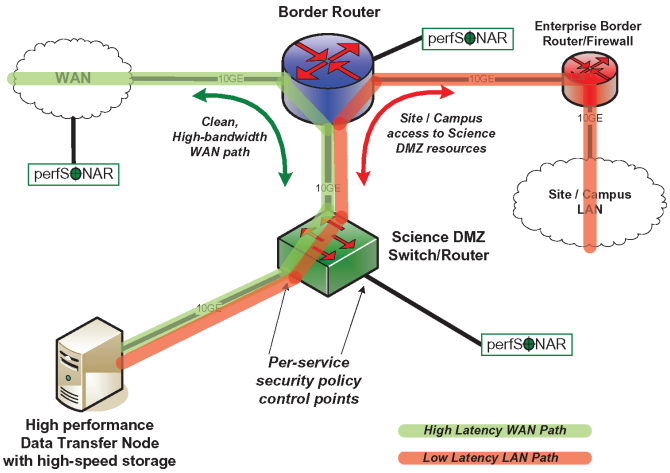


Fig. 1: A typical Science DMZ (from [10]).

is the data transfer node (DTN), servers provisioned for and dedicated to efficient high-speed data transfers between sites. As dedicated systems, DTNs do not have general computing applications installed (e.g. email clients, media players, document editors) [2], and are limited to parallel data transfers tools such as GridFTP, and performance monitoring tools such as perfSONAR [11] [2] [12].

III. System Performance Metrics

Predictions about the network activity can be made by considering the system metrics. For example, increased network I/O is associated with increased CPU usage [13]. Previous work has demonstrated that certain features can be used to identify different types of network traffic, including attacks such as SYN floods or port scanning [5]. In particular, the relationship between SYN floods and significant increases in interrupts has been demonstrated [6]. Interrupts signal to the CPU that I/O needs to be performed, and each new SYN request triggers a new interrupt. A large number of interrupts, coupled with an unexpectedly small increase in other metrics that indicate meaningful work, such as context switches, can be used to identify flooding attacks [6].

Because DTNs perform a relatively narrow range of network activities, they generate predictable network and disk I/O activity [2] [10], and make them ideal environments for host-based anomaly detection. Therefore, we believe that SYN floods and other attacks can be detected through anomalous patterns of system metrics. The relationship between system metrics such as interrupts and the network metrics such as packets received can also be combined to enhance detection. For example, abnormal relationships between the interrupts per second and packets or bytes received per second could potentially identify malicious traffic. We focus primarily on three different per-second metrics in our experiments - interrupts, packets received, and context switches.

IV. Machine Learning for Anomaly Detection

Machine learning algorithms, including clustering, excel at finding pattern similarities and are thus classically applied to predictive classification problems [14]. The anomaly detection problem (also known as the outlier problem) seeks to find *new* patterns in data that do not conform to expected behavior [15] [16]. It too can be modeled as a classification problem with "normal" and "abnormal" as classes. Sommer and Paxson [17] argue that the application of machine learning for anomaly detection must be used with care when using a "closed world assumption." Witten et al. [18] defined a closed world assumption as: *The idea of specifying only positive examples and adopting a standing assumption that the rest are negative is called the closed world assumption*. We address some of those points of concern here:

- *Bridging the Semantic gap*: Anomalous activity does not necessarily translate to an attack, and anomaly detection might be better suited for known attacks over novel ones. In our experiments, we carried out both normal baseline activity and the attacks so that the ground truth is known.
- *Narrow Scope*: DTNs have limited system functionality, as discussed in Section II, resulting in predictable network and system activity. This context makes it suitable for anomaly detection via machine learning.
- *Real Data*: We generated our own data using a test DTN and emulated scientific workflows with other test DTNs (as we were unable to obtain campus-level DTN data).

Anomaly detection systems often give critical, actionable information, so a good system should provide this information as soon as possible with the ability to predict anomalies in real-time with streaming data. DBSCAN clustering is one such an unsupervised machine learning approach that exhibits properties ideal for anomaly detection with noisy data streams in real-time [19]. The next section gives a broad overview of the algorithm and its derivative anomaly detection system.

V. DBSCAN

DBSCAN, short for density-based spatial clustering of applications with noise, is a well-established clustering algorithm [8]. Clusters are grouped together based on the density of the points, with low density points labeled as noise. Unlike k-means clustering, DBSCAN does not require you to specify the number of clusters beforehand, and the clusters can be any shape. DBSCAN only requires two parameters, ϵ (the greatest distance allowed between points in each cluster) and **MinPts** (the minimum number of points required for a cluster). Clustering algorithms like DBSCAN are a form of unsupervised learning, meaning learns without the need for training nor labeled data, both of which are needed with supervised approaches. This makes it a strong choice for monitoring real-time data, since DBSCAN allows us to differentiate between normal patterns and anomalous ones despite not knowing the exact thresholds beforehand. In addition, DBSCAN is designed for clustering noisy data [19], which makes it well-suited for monitoring the sometimes unpredictable network and system performance metrics. These

properties make DBSCAN a good candidate for our anomaly detection system.

VI. Experimental Setup

For our experiments, we set-up and monitored a machine, denoted as D, to act as a Science DMZ DTN, like that shown in Figure 1. D is a PowerEdge T630 server with a 10 Gbps NIC, two Intel Xeon E5-2637 v3 3.5GHz processors, a RAID-10 set of 8 1TB 7.2K RPM SATA 6 Gbps hard drives, and 32GB 2133MT/s RDIMM memory capacity. D has a 10 Gbps backbone link to CENIC, a 100 Gbps wide-area network, as is true for our campus DTN. As in a true Science DMZ [2], this machine requests data from other DTNs and receives files through the 10 Gbps link using Globus GridFTP [12] transfers. We emulate real scientific workloads by having D receive test data via GridFTP transfers from three read-only test DTNs provided by ESnet [20]. For each day of experiments, we continuously receive randomly selected files from one of these three test DTNs at randomized time intervals. The potential file sizes are shown in table I. The actual workload of a DTN varies significantly depending on the organization and type of scientific workflow. Therefore we also consider two additional cases - a large number of small file transfers, and a small number of large file transfers. This simulates the different distributions that have been observed on a real NERSC DTN [21]. In each case, the time between file transfers has been chosen such that the total expected data transferred is roughly 750 GB per day.

TABLE I: GridFTP transfer distributions

Distribution	Potential File Sizes	Interval
Normal	10M, 50M, 100M, 1G, 10G, 50G	1-30 minutes
Large	10G, 50G	60-75 minutes
Small	10M, 50M, 100M, 1G	5-40 seconds

In addition to D, we have a machine designated as the attacker, denoted as A, with the same specifications as D. A is connected to D by a 40 Gbps Mellanox ConnectX-3 network adapter. To generate our TCP SYN flood attacks, we use hping3 [22], a freely available packet generator used for penetration testing. Using hping3, we have A generate SYN packets and send them to D over the 40 Gbps link. In order to determine the point at which unusual patterns of SYN packets becomes detectable, we vary the attack intensity by increasing or decreasing the number of microseconds between SYN packets sent. Each attack lasts for 15 minutes, with the intensity gradually increasing by reducing the delay between packets by 5 μs every 90 seconds. At the max intensity, hping3 just sends a flood of SYN packets with no delays in-between for 15 minutes. Six different intensities, with varying inter-packet delay ranges, are used in our experiments – **lowest** (150-100 μs), **low** (125-75 μs), **medium** (100-50 μs), **high** (75-25 μs), **higher** (50-0 μs), and **max** (0 μs). We performed four experiments using this system configuration and set-up as shown in figure 3. Three of the experiments involved periodic SYN floods occurring every two hours starting at noon, with each intensity level used once. The final experiment sends the medium, higher, and max intensity floods once, each at random times during the day. While our experiments run, we log system (interrupts, context switches) and network data (packets received, bytes received) per second on D using the `proc` filesystem.

Finally, we use DBSCAN clustering [8] to detect the anomalies in each time-series performance data set collected. First, we used DBSCAN to cluster the 24 hours worth of logged performance data gathered for each of the experiments shown in figure 3. The standard method of detecting TCP-SYN floods is to check whether or not the number of SYN packets received per second exceeds a predetermined threshold [6]. As figure 2 shows, looking at the packets received

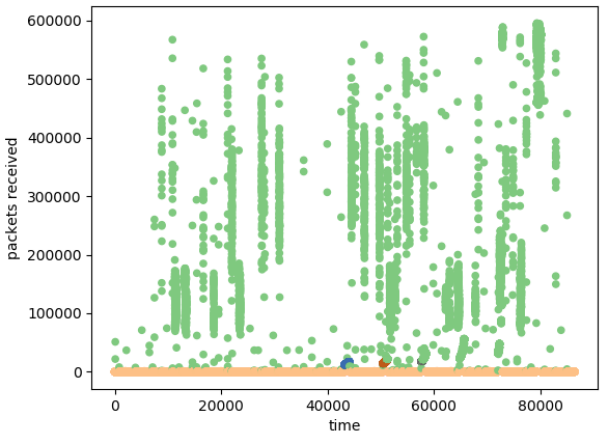
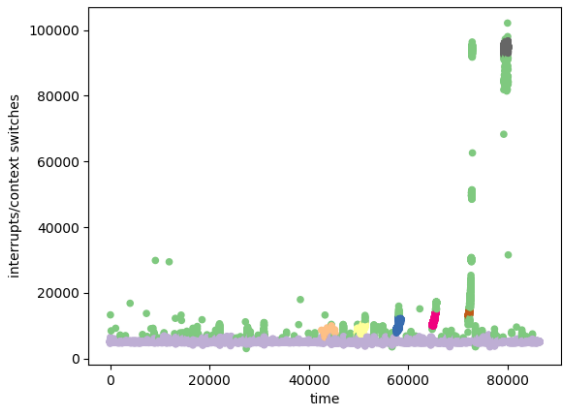


Fig. 2: DBSCAN clustering using packets received per second on the DTN. The SYN floods here are generally indistinguishable based on packet volume alone.

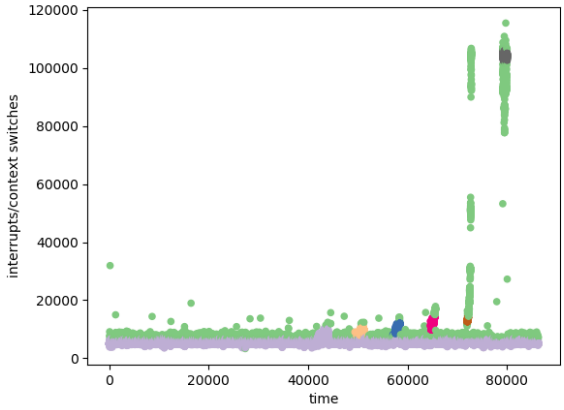
alone is likely insufficient on the DTN, particularly for low volume attacks, as file transfers can be mistaken for TCP-SYN floods. Instead, we consider the results of clustering interrupts over context switches vs. time for each of the four experiments over the entire day. We set the parameters such that, under normal conditions, there will only be a single cluster representing the normal range of interrupts over context switches. Therefore, if another cluster is discovered, it indicates a period of time where there is a high density of abnormal behavior. Then, we consider the effectiveness of a real-time detection system using DBSCAN clustering. The real-time streaming DBSCAN detection system continually adds data from the `proc` filesystem to a `pandas` dataframe. Once the dataframe grows large enough for clustering, we run DBSCAN and count the number of resulting clusters to check for anomalies. We consider the time required to detect SYN floods of different rates, and trade-offs associated with the various parameters.

VII. Evaluation and Discussion

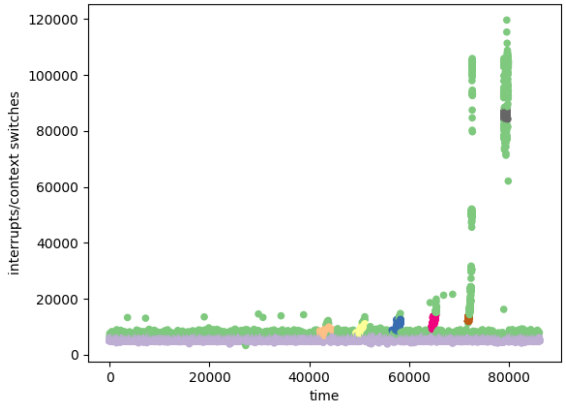
In figure 3, we show the results of DBSCAN clustering on 24 hour DTN experiments. In these figures, the interrupts over context switches values have been normalized by multiplying



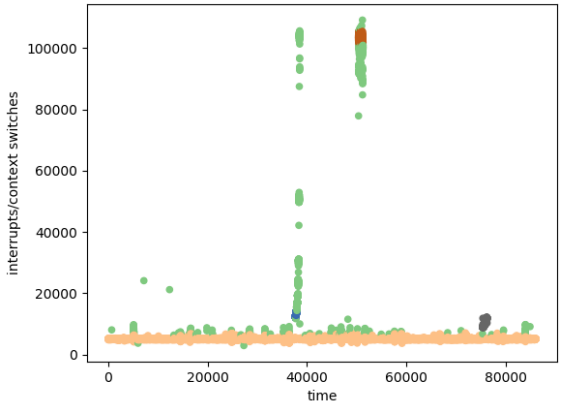
(a) Experiment 1 (periodic floods, random file transfers)



(b) Experiment 2 (periodic floods, large file transfers)



(c) Experiment 3 (periodic floods, small file transfers)



(d) Experiment 4 (random floods, random file transfers)

Fig. 3: DBSCAN clustering results for the normalized interrupts over context switches vs. time (in seconds). In each subfigure, the largest cluster represents standard activity, while the green dots represent un-clustered noise. The other small clusters occur during SYN floods. In all cases, the SYN floods can be distinguished from ordinary traffic.

them by 10,000 to improve the readability of the charts and simplify the choice of the DBSCAN distance parameter ϵ. To obtain these figures, we set ϵ to 910, and **MinPts** to 200. During ordinary conditions on the DTN (file transfers or inactivity), we found that the interrupts over context switches remained fairly consistent around 0.5 (5000 in the figure 3), with some noise occurring throughout the day. However, ordinary spikes occur infrequently, compared to densely packed clusters which appear during the attacks. The small clusters shown in figure 3 occur only during the hping3 SYN floods. A large number of interrupts coinciding with a relatively low number of context switches generally indicates that less meaningful work is being performed [6]. Therefore, it is sensible that this is a good indicator of SYN floods, where a large amount of "useless" packets are handled within a short time frame. As expected, the higher intensity floods showed significantly higher interrupt over context switch ratios.

A. *Real-time Anomaly Detection*

Following the analysis of the full days worth of traffic, we consider how DBSCAN could be applied to detect attacks in real-time. A streaming anomaly detection system was created using Python, which gathers performance data and creates new clusters every 10 seconds. If an unexpected number of clusters is detected during a time slice, then an anomaly is reported. In our experiments, we found that accurate detection could be performed even clustering a small number of points. The recommended number for DBSCAN's **MinPts** parameter is one more than the number of dimensions [23]. In our case, the number of dimensions is two – the interrupts over context switches ratio and the time. Since there are only two dimensions, we don't need a large sample to perform reliable clustering. We found that the minimum recommended value of three points per cluster was sufficient to detect the attacks without false positives. Random spikes caused by noise never occurred closely enough together in a short time

frame to trigger false alerts. The minimum time for detection is determined by how many points we choose to gather before re-clustering. Our experiments used 10 data points, making 10 seconds the minimum detection time. Figure 4 shows that most attacks were detected in 10 seconds using the minimum number of points, even at very low rates. In our experiments, we managed to detect SYN flooding at as low as 4,800 packets per second. Although we detected all attacks with no false positives, the bandwidth measuring tool iperf produces a similarly high interrupts over context switch ratio. As a common application used to test DTN performance [1], this could result in false positives. However, even if the scheduled iperf tests are not recognized beforehand, we can eliminate any false positives by also considering the number of bytes received per second in our analysis. We found that the bytes per second during an iperf test is at least one order of magnitude higher than during the SYN floods we tested. Therefore, we can improve detection further by combining network and performance metrics.

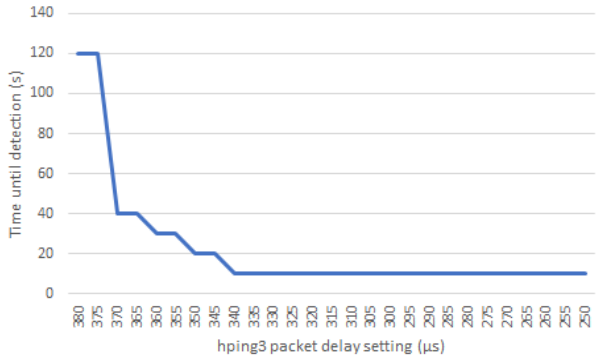


Fig. 4: Time spent before detection during low intensity TCP-SYN floods. Over the course of an attack, the number of interrupts gradually increases, until it becomes detectable. All but the lowest rate attacks were detected after the minimum 10 seconds required to gather data for clustering.

VIII. Conclusions and Future Work

Our work demonstrates that system performance data can be effective in detecting TCP-SYN floods, an attack traditionally detected by a network IDS, on a Science DMZ DTN. The limited variety of tasks typically performed on a DTN helps us use system performance metrics to reliably detect threats. Interrupts over context switches consistently detected the SYN floods, even at very low intensities. Combining this with network metrics such as the number of packets or bytes received per second can improve detection. DBSCAN can applied in a real-time detection system, clustering performance metrics over time to detect attacks. Successfully detecting SYN floods on the DTN suggests that this system could be generalized for other threats, such as port scanning or insider attacks which are unlikely to be detected with network activity alone. With that in mind, these metrics could be valuable being incorporated into an intrusion detection system such as Zeek (Bro). Future work could also involve using both network and system performance metrics (e.g. context switches, etc.) to detect insider attack scenarios on a DTN.

References

[1] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on Science DMZ," *IEEE Communications Surveys & Tutorials*, 2018.

[2] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science dmz: A network design pattern for data-intensive science," *Scientific Programming*, vol. 22, no. 2, pp. 173–185, 2014.

[3] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[4] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks.," in *Lisa*, vol. 99, pp. 229–238, 1999.

[5] A. Visconti, N. Fusi, and H. Tahayori, "Intrusion detection via artificial immune system: A performance-based approach," in *IFIP International Conference on Biologically Inspired Collaborative Computing*, pp. 125–135, Springer, 2008.

[6] A. Aqil, A. O. Atya, T. Jaeger, S. V. Krishnamurthy, K. Levitt, P. D. McDaniel, J. Rowe, and A. Swami, "Detection of stealthy TCP-based DoS attacks," in *MILCOM 2015-2015 IEEE Military Communications Conference*, pp. 348–353, IEEE, 2015.

[7] E. Lawrence Berkeley National Lab, "DTN tuning," 2019.

[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[9] N. S. Foundation, "Campus cyberinfrastructure (cc*) program solicitation," 2019.

[10] S. Peisert, W. Barnett, E. Dart, J. Cuff, R. L. Grossman, E. Balas, A. Berman, A. Shankar, and B. Tierney, "The medical Science DMZ," *Journal of the American Medical Informatics Association*, vol. 23, no. 6, pp. 1199–1201, 2016.

[11] A. Hanemann, J. W. Boote, E. L. Boyd, J. Durand, L. Kudarimoti, R. Łapacz, D. M. Swany, S. Trocha, and J. Zurawski, "Perfsonar: A service oriented architecture for multi-domain network monitoring," in *International conference on service-oriented computing*, pp. 241–254, Springer, 2005.

[12] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The globus striped gridFTP framework and server," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, p. 54, IEEE Computer Society, 2005.

[13] T. Park, Y. Kim, J. Park, H. Suh, B. Hong, and S. Shin, "Qose: Quality of security a network security framework with distributed nfv," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2016.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[15] V. Chandola, V. Mithal, and V. Kumar, "Comparative evaluation of anomaly detection techniques for sequence data," in *2008 Eighth IEEE international conference on data mining*, pp. 743–748, IEEE, 2008.

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[17] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*, pp. 305–316, IEEE, 2010.

[18] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[19] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, p. 19, 2017.

[20] ESnet, "ESnet," 2019.

[21] A. Giannakou, D. Gunter, and S. Peisert, "Flowzilla: A methodology for detecting data transfer anomalies in research networks," in *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, pp. 1–9, IEEE, 2018.

[22] K. Tools, "hping3. ICMP or SYN flooding tool," 2014.

[23] M. Hahsler, M. Piekenbrock, and D. Doran, "dbscan: Fast density-based clustering with r," *Journal of Statistical Software*, vol. 25, pp. 409–416.