Next-Generation Intrusion Detection and Prevention System Performance in Distributed Big Data Network Security Architectures

Michael Hart[1], Rushit Dave[2], Eric Richardson[3]
College of Science, Engineering, & Technology, Minnesota State University, Mankato, United States[1, 2]
College of Health and Human Services, University of North Carolina Wilmington, United States[3]

***Abstract*—Big data systems are expanding to support the rapidly growing needs of massive scale data analytics. To safeguard user data, the design and placement of cybersecurity systems is also evolving as organizations to increase their big data portfolios. One of several challenges presented by these changes is benchmarking real-time big data systems that use different network security architectures. This work introduces an eight-step benchmark process to evaluate big data systems in varying architectural environments. The benchmark is tested on real-time big data systems running in perimeter-based and perimeter-less network environments. Findings show that marginal I/O differences exist on distributed file systems between network architectures. However, during various types of cyber incidents such as distributed denial of service (DDoS) attacks, certain security architectures like zero trust require more system resources than perimeter-based architectures. Results illustrate the need to broaden research on optimal benchmarking and security approaches for massive scale distributed computing systems.**

***Keywords*—Big data systems; zero trust architecture; benchmarking; distributed denial of service attacks**

I. Introduction

Big data systems are unified environments designed for massive-scale data analytics. Systems capable of handling large amounts of data are becoming more important as the volume of data created and communicated over the Internet increases [1]. Cybersecurity systems play an important role in ensuring the large quantities of data on the Internet remains safe. One dimension of several necessary to accomplish the latter are next-generation security devices. Intrusion detection and prevention systems (IDPSs) properly manage data accessibility, privacy, and safety. IDPS algorithms are able to identify cyber threats using several mechanisms. This includes using prior information from previous attacks, anomalies in network packets [1], and machine learning [2].

As big data systems become more common, their roles will continue to expand. This includes the capability to analyze and detect information security vulnerabilities at scale. For example, several big data frameworks exist that discover distributed denial of service (DDoS) attacks [3]. This expansion of roles offers many exciting opportunities for organizations. However, as the use of big data systems grows, the capability of attackers to leverage associated parallel computing power for nefarious reasons also increases [3]. A systematic review of 32 papers pertaining to securing big data found that a critical need in future research is building more secure big data infrastructure [4]. Contributing to the latter objective, the researchers demonstrate how varying network architectures impact the security and performance of big data systems.

Organization of the paper is as follows. Section II reviews literature on intrusion detection and prevention methods for big data systems. Section III outlines the research design and methodologies used to test perimeter-based security and perimeter-less security applied to a big data system environment. Section IV describes the research results. Section V concludes the study by discussing the limitations and future outlook.

II. Literature Review

Work is necessary to optimize both the information security and performance of distributed systems. Today, several open-source big data frameworks provide remarkable potential for solving challenging data science and related problems by leveraging powerful parallel and distributed data processing. However, securing these systems often carries performance penalties. The review of literature that follows explores research on the impact of various IT infrastructure security strategies and their influence on big data environments. It begins by reviewing comprehensive surveys most closely related to information security and big data systems.

A. Surveys of Big Data and Intrustion Detection

Previous systematic reviews of literature focused on information security and big data provide a vast array of objectives. A prominent theme is using deep learning [1] and machine learning [2] to assist in detecting or preventing cybersecurity attacks. This line of research often utilizes deep learning or machine learning algorithms for near real-time data protection.

A recent and well cited comprehensive survey in [1] evaluates how deep learning is used for intrusion detection systems in the cybersecurity domain. It found notable contrasts between machine learning approaches in cybersecurity and deep learning. Conventional machine learning approaches utilized in cybersecurity were classified by approaches such as artificial neural networks (ANNs), Bayesian networks, decision trees, fuzzy logic, k-means clustering, k-nearest neighbor (kNN) algorithm, and support vector machines (SVMs). The

survey centered on deep learning focal intrusion detection methods that included autoencoders (AEs), convolutional neural networks (CNNs), deep belief networks (DBNs), generative adversarial networks (GANs), and long short-term memory (LSTM) recurrent neural networks [1].

AEs, DBNs, and GANs were highlighted in [1] for their unsupervised learning strengths. In the absence of gradient estimation, AEs can use gradient descent to train data. A strength of LSTM is its capabilities in analyzing time-series data. CNNs do not need as much data processing prior to evaluation as certain algorithms and is able to classify cyber-attacks using multiple characteristics well. Combined, the survey of literature finds that AEs, CNNs, DBNs, GANs, and LSTM networks each have potential to improve intrusion detection methods. Furthermore, the survey [1] outlined the importance of dataset reliability when evaluating deep learning intrusion detection effectiveness. Variance in cybersecurity attack datasets can introduce model bias when comparing multiple deep learning methods. Thus, any biases in attack datasets or data from live systems could increase spurious results [1].

A subsequent theme in the literature concentrates on cybersecurity and privacy prevention in big data applications. While this research again employs various data science methods to detect or prevent data breaches, it also illustrates how big data techniques can prevent information privacy issues. Research in [4] led to a proposed model for enhancing information privacy. The model highlights people, organizations, society, and government roles. It leverages IDS, IPS, and encryption as its primary techniques to prevent data breaches [4].

B. Big Data Architectures and Information Security

As big data evolves, the supporting infrastructures will require proper encryption, intrusion detection, and intrusion prevention. Changing architectures within computer networks, messaging techniques, and undefined communication methods introduce numerous challenges. In a 2014 study Mitchel and Chen [5] recognized this paradigm. Their emphasis on cyber-physical systems (CPS) ranging from smart grids to unmanned aircraft systems led to the classification of four primary intrusion detection categories. These include legacy technologies, attack sophistication, closed control loops, and physical process monitoring. Each of the latter is narrow concepts as they relate to the broader field of intrusion detection, underlying the unique customization of IDSs for cyber-physical systems [5].

Three years later Zarpelo et al. [6] outlined a similar but distinct paradigm; intrusion detection focal to the Internet of things (IoT). The researchers stated that IoT has similar information security matters as the Internet, cloud services, and wireless sensor networks (WSNs). Despite similarities, IoT information security approaches are distinct, according to the authors due to concepts such as data sharing between users, the volume of interconnected objects, and the amount of computational power of the associated devices. Like cyber-physical systems, IoT presents diverse challenges to the design of instruction detection systems [6].

Designing secure cloud computing environments poses several novel problems at multiple infrastructure layers. As an example, cloud resources can be leased by numerous vendors focused on varying as-a-service models such as infrastructure as a service (Iaas), platform as a service (PaaS), and/or software as a service (SaaS). Multi-cloud applications rely upon the seamless integration of cloud resources from providers focused on one or many as-a-service types, which continue to expand. In Casola et al. [7] a model is outlined for designing, creating, and implementing multi-cloud applications. The flexible approach accounts for varying as-a-service components. Security-by-design is a primary objective of the process lifecycle between the functional design of multi-cloud applications and the security design. The functional design phase defines the application logic, interconnections of services, and resource requirements. In the security design phase, each cloud element is assessed in terms of security risks and security needs. Security policies and controls are designed based on the latter requirements. Similar to CPS [5] and IoT [6], the multi-cloud application model is a subsequent example of how information security solutions play a prominent role due to the systems' distinct architectural and infrastructure layers.

Securing big data environments or leveraging associated techniques like machine learning to enhance information security intertwines numerous fields include but not limited to CPS, IoT, and cloud computing. Like big data systems, CPS requires cybersecurity protection [8] of private data [9]. Big data, IoT, and CPS often overlap through the ad hoc interfaces of systems such as smart vehicles, buildings, factories, transportation systems, and grids [10]. As a vulnerable attack surface, IoT advances the need for intelligent information security.

Machine learning [11], including ensemble intrusion detection [12], and IDS design [13] are proposed techniques to mitigate malicious cybersecurity attacks. Due in part to porous attack surfaces in cloud centric big data, IDSs may require collaborative frameworks [14]. In [15], fuzzy c means cluster (FCM) and support vector machine (SVM) were proposed as a collaborative technique for IDS detection rates. Compared to other mechanisms, the proposed hybrid FCM-SVM showed lower false alarm ratios and higher detection accuracy [15]. Furthermore, [16] illuminates the need for scaling IDS detection algorithms using the resources of parallel computing in the cloud.

In [17] the researchers propose the BigCloud security-by-design framework. The framework draws from the need to integrate big data security into the system development lifecycle. Its primary cloud application domain is focal to infrastructure as a service. It notes IaaS as one of the faster growing as-a-service options for big data. The model helps design and enforce secure authentication, authorization, data auditability, availability, confidentiality, integrity, and privacy. However, its IaaS concentration could provide greater benefits to as-a-service components specific to host operating systems, hypervisors, networking, and hardware [17]. Similar to IaaS, the evolution of serverless platforms and Function-as-a-service (FaaS) applications requires careful security design to overcome security threats that new services often suffer [18].

While distinct, CPS, IoT, cloud computing, and big data are merely a few examples of why designing intrusion detection and prevention systems remains highly elastic in modern computational architectures. As the information technology landscape changes, information security bends to meet the evolving needs of the complete environment. To conclude the literature review, the authors will outline several relevant studies introducing potential solutions to design stronger information security controls for big data systems.

C. Encryption

An ongoing challenge in distributed big data systems is securing communication between multiple systems operating across various computer networks. Apache Hadoop and Apache Spark are examples of big data frameworks that present several opportunities for attackers to access the data they facilitate. Central to big data frameworks is the ability to use parallel processing to analyze massive amounts of data. MapReduce is one of many programming paradigms that leverages Hadoop to extract valuable knowledge from large volumes of data. However, like most application or service modules within big data frameworks, MapReduce highlights the vast attack vectors that exist in distributed big data systems. MapReduce examples in literature include side channel attacks [19], job composition attacks [20], and malicious worker compromises in the form of distributed denial-of-service (DDoS) or replay attacks [21], Eaves dropping and data tampering [22]. Encryption is a primary countermeasure to secure transmissions and prevent data leaks between big data servers [19].

A primary objective in addressing cybersecurity attacks on parallel processing services is identifying and preventing leaks that often occur during data transmission between distributed worker nodes, also referred to as DataNodes in Apache Hadoop. These unique yet integrated servers work in parallel to complete MapReduce jobs. Often in Hadoop, data is stored and retrieved from the Hadoop Distributed File System (HDFS). In [19] side-channel attacks are addressed that can occur between MapReduce workers that utilize HDFS for data storage. These types of cybersecurity attacks can target worker nodes to extract valuable information pertaining to MapReduce jobs such as the amount of packet bandwidth. This further contributes to successful pattern attacks. The authors proposed a solution to this vulnerability labeled Strong Shuffle that enforces strong data hiding between workers [19]. In contrast to alternative countermeasures such as correlation hiding in [20], Strong Shuffle avoids leaking the number of records accepted by each reducer during MapReduce runtime. Secure plaintext communications is a function of semantically secure encryption in the Strong Shuffle solution [19].

In [19] data communicated between Hadoop DataNodes and stored in HDFS is encrypted with semantically secure AES-128-GCM encryption. Although the latter helps prevent clear text leakage between MapReduce jobs in Hadoop, encryption in big data environments has limitations. For example, encrypted databases can still reveal certain information during operations that include table queries. Deterministic encryption and order-preserving encryption can leak the equality relationship and the order between records. One proposed solution is semantically secure encryption. In [23] the authors propose a semantically secure database system named Arx. Alternative to order-preserving encryption, semantic security within Arx only allows an attacker to extract order relationships and frequency of the direct database query in use in contrast to the entire database. The authors note that worst-case attackers would gain as much information from a data leak as deterministic or order-preserving encryption over time [23]. While methods such as encryption and authentication help with cross-node data leaks, they do not prevent other attacks, such as DDoS and passive network eavesdropping [21]. A subsequent countermeasure is the effective design and implementation of intrusion detection and prevention systems [14].

D. Next-Generation Security and Big Data Systems

Next-generation security at a high level can detect and prevent malicious cybersecurity attacks. Much of the literature focuses on identifying malicious network packets in real-time. The comprehensive survey in [24] reviews how modern data mining techniques are evolving to meet real-time detection needs. The review classifies intrusion detection systems by architecture, implementation, and detection methods. Detection methods are categorized as anomaly-based, signature based, and hybrids. Signature based methods or misuse often rely upon a database that defines patterns or existing malicious attack signatures. Anomaly detection can detect non-normal network traffic behavior that has yet to be defined in a signature database. Data mining methods including supervised, unsupervised, and hybrid learning are being used to improve anomaly-based intrusion detection systems [24].

While supervised, unsupervised, and hybrid learning IDS research continues to progress [24], the ongoing need to improve existing big data implementations remains. In several systematic literature reviews [1, 2, 3, 24], IDSs are known to have limitations that contradict the performance benefits of parallel processing and distributed computing. For example, large signature based systems drain CPU and memory resources [24]. While researchers continue to advance areas of intrusion detection such as packet anomalies and encryption, only a few studies are advancing security by design and its effects on varying big data architectures [1]. To address this need, the authors of this study designed a distributed big data system over a wide area network to explore the performance of distributed nodes under different network traffic loads.

III. METHODS

This research methodology follows the design science approach in [25] and [26]. Design science is based on a scientific framework for IT research. As March and Smith [25] outline, IT research should consider natural and design science as a method to build and evaluate tangible objects. Within this philosophy, objects often have outputs in the form of models or instantiations. Instantiations associate with new artifacts in the design science methodology and the understanding of the artifact in its environment [25]. IT artifacts can be realized in many forms such as through the design of an object that helps solve business problems [26].

A. Organizational Problem

Central to the organizational problem in this study is the need to architect a real-world or simulated big data environment that generates important inputs and outputs. In the case of this study, several architectural layers require design, configuration, benchmarking, and evaluation that accurately represent industry big data system implementations. These research activities could establish a more mature model for IDPS placement in evolving network architectures. Design science methods guide the latter activities [26].

Big data clusters can have thousands of nodes. Attempting to secure individual servers poses several issues ranging from significant costs to lost computational resources. Important to the artifact design process is the creation of an IDS and IPS testing environment that results in minimal disruption to existing big data infrastructures. Additionally, the authors constructed an experimental setup similar to several local small business environments that are readily available, relatively inexpensive, and relevant to a broad audience. Therefore, the testing environment is limited to several small commodity virtual machines (VMs) operating in physically distanced data centers. The authors will briefly outline the network architecture, hardware, software used in the experimental environment.

B. Network Architecture

Fig. 1 depicts the baseline network architecture used in this study. The experimental network emulates a small to medium-sized business with a 200 Mbps dedicated lease line between four distinct physical locations. Connections are 1 Gbps copper from the demarcation point to the LAN nodes. Each server is connected to layer 2 switches followed by a layer 3 Cisco Systems enterprise class router.

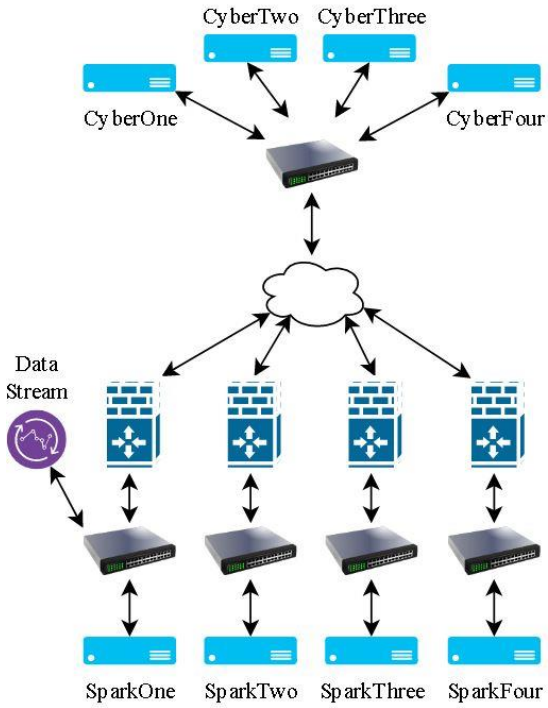

Fig. 1. Perimeter-based security network architecture.

The cybersecurity servers labeled “CyberOne” to “CyberFour” illustrate the systems used to attack the big data cluster. The big data cluster includes four servers labeled “SparkOne” to “SparkFour.” One streaming server is depicted as the data stream located in the same local area network (LAN) as SparkOne. Four intrusion detection and prevention systems are situated between each big data server and its extrinsic networks.

C. Hardware

The big data servers run on parallel Dell hardware [27]. The hardware is manufactured on the same date and shipped in the same container. The testing server used the same single Intel CPU with 16 logical cores and 32 GBs of physical random-access memory. The baseline Intel CPU benchmark average results from the PassMark version 10 performance test [29] are 2,799 MOps per second for a single thread and 5,443 megabytes per second for data encryption.

Cisco RV series routers with integrated firewalls exist between each Apache Spark node and the external network. Cisco Firmware 1.0.3.55 is in use with the default firewall ruleset. The authors added customized rules that allow the internal LAN IP addresses to communicate on the necessary Apache HDFS and Spark ports. Subsequent ports are blocked [28].

D. Big Data Systems

Each big data server and streaming server used equivalent software and versions. Systems ran on the Ubuntu server 20.04.3 LTS operating system. Installed software included Java 11, Python 3.8, Apache Hadoop 3.2, and Apache Spark 3.2. The big data environment is comprised of five servers. This includes one primary cluster manager labeled *SparkOne* and three secondary work nodes labeled *SparkTwo*, *SparkThree*, and *SparkFour*. Apache Spark is tuned using optimal parameters such as those specified in [30] and [31]. HDFS disks are balanced between nodes with DFS replicating three blocks. The data stream denotes the independent Spark streaming instance.

SparkOne is the primary node in the testing environment used in this study. It is comprised of the driver program. The driver program executes the big data application’s main() class and generates the SparkContext [32]. SparkContext is capable of using various big data resource managers. Tests in this study use Yet Another Resource Negotiator (YARN) as the distributed cluster manager [33].

SparkContext helps communicate application jobs containing code in various forms such as Python and JAR files to the executors on the worker or secondary nodes in the cluster. YARN has two primary high-level components labeled the NodeManager and ResourceManager. Secondary nodes in a big data cluster managed by YARN each have a NodeManager. Its function is to manage containers on each server. Containers encompass resources such as network, disk, CPU, and memory. These are allocated properly to facilitate task execution. The YARN ResourceManager consists of the ApplicationsManager and the Scheduler. While the Scheduler determines the necessary resources for each application the

ApplicationsManager identifies which container the application will use and subsequently monitors their task execution [33].

Apache Spark and HDFS replicate between three secondary big data servers. The secondary or worker nodes labeled SparkTwo, SparkThree, and SparkFour contain executor processes. An executor process remains throughout the runtime of tasks that each worker is allocated by the cluster manager. Every application receives its own executor process and/or processes as necessary. The driver program on SparkOne is configured to listen for executor process communications from the secondary nodes until the job is completed. Per Apache Spark documentation in [32], when possible, the driver program should be on the same local area network as the worker nodes due to the latter communication. In the experimental network design, the worker nodes are physically distanced. Therefore, Spark is optimized to open local remote procedure calls on the worker LANs [32].

E. Attack Systems

Although the cybersecurity servers ran on the same hardware as the big data servers, they used different software. CyberOne, CyberTwo, CyberThree, and CyberFour each delineate a server used to carry out cyber-attacks on the big data cluster. The software includes the Kali Linux operating system running the 5.14 kernel. Kali Linux is an open-source operating system based on Debian Linux. It is designed for numerous information security objectives such as reverse engineering, forensics, pen testing, and research [34].

F. Intrustion Detection and Prevention Systems

Consistent with Fig. 1, the baseline IDS and IPS systems are located between the cyber-attack and big data systems. Regardless, the authors manipulate the placement of these systems throughout each experimentation. As a simulated construct in the research methodology, the authors propose that IDS and IPS architecture placement predicts data streaming performance between worker nodes. Performance evaluation of this potential construct is an important step toward advancing a future IDPS placement framework for physically distanced big data systems.

The authors implemented Snort and Suricata, two popular open-source IDS and IPS systems. Snort is developed by Cisco Systems. It serves as a leading intrusion detection engine and rule set for Cisco next-generation firewalls and IPSs. Its mechanisms for detecting and preventing security threats continue to evolve. However, a fundamental capability during this writing is the formation of rules. In contrast to traditional methods such as signature-based detection, rules focus on vulnerability detection [35]. Suricata is developed by the Open Information Security Foundation (OISF). Similar to Snort, Suricata can use rules to detect and block cyber-attacks [36].

Version 2.9.7 of Snort ran with libpcap version 1.9.1 and version 8.39 of the payload detection rules. Suricata testing uses version 6.0.6 with the emerging threats open ruleset. The authors customized the latter default Snort and Suricata rulesets to secure the distributed nodes. The rulesets are parallel in count and type (e.g. alert, drop) to control significant variations in resource contention. Suricata and Snort use the same rules in the tests, except for minor incompatibilities. Where incompatible, the rules are adjusted to perform the same action in both IDSs at parallel throughput rates.

Snort and Suricata run on the same server hardware and operating systems as the big data servers. A second NIC allows the servers to act as gateways between trusted and untrusted networks. The servers communicate between the local area networks using Transport Layer Security (TLS) and Secure Shell (SSH) Protocols. Ubuntu server 20.04.3 LTS is configured using OpenSSH version 8.2 and OpenSSL version 1.1.1.

G. Benchmarks

The authors developed custom benchmarks to identify how big data clusters perform under various IDS physically distanced network architectures. The benchmarks perform two significant network load functions, 1) streaming unstructured data to the Spark big data cluster and 2) flooding the Spark nodes via DDoS attacks. Network and system benchmarking uses version 16m of the nmon source code to measure network performance. Originally developed by IBM, nmon is an open-source Linux project that monitors system resource utilization. Performance metrics include CPU, disk, memory, and networking [37].

The authors follow the design science methodology [25] to design and implement an IDS placement experiment for physically distanced big data systems. Next, the authors construct a series of tests to determine how IDS locations influence real-world distributed worker nodes.

IV. RESULTS

Each of the tests followed an eight-step process, 1) network architecture is determined and implemented, 2) IDPS locations are identified and configured, 3) IDPS customized rulesets are implemented, 4) the big data system cluster is started and tested as operational, 5) data streams to the cluster are invoked, 6) DDoS attacks are executed, 7) the benchmarks are run, and 8) the researchers maintain and monitor the testing environment for anomalies. Each of the tests was repeated three times to ensure saturation existed in the results.

A. Test 1 Perimeter-Based Security Results

Fig. 1 illustrates the IDPS placement location for the first test. The cloud represents the leased line between the geographical sites. Below the cloud icon is the selected IDPS solution followed by the Apache Spark cluster. Network architecture in the first test follows Cisco Systems' best practices for a collapsed data center and LAN core [38]. Within this design, a hardware-based IDPS is situated between the public untrusted and private trusted networks. Test one includes a traditional perimeter Cisco Systems IDPS. Individual Spark nodes are networked in a single VLAN connected through the collapsed core.

In contrast to the network architecture in Fig. 1, CyberOne through CyberFour servers are not deployed for tests 1-3. In each of these tests, typical network traffic is present void of any DDoS attacks.

Benchmark metrics are specific to the big data systems unless otherwise specified. During the data stream, HDFS is

writing 128 MB blocks to disk on all three Spark worker nodes at a constant rate. Inconsequential wait time exists on disk reads and writes. Average CPU utilization per thread or "CPU%" on the big data worker nodes is 4.3% during the first test. The average time a process waits for an input-output (I/O) to complete or "wait%" is 0.3. The average number of processor context switches per second is 1,728, identified as "PWps" hereafter.

The authors measured network performance between each of the Spark nodes using four metrics. Metrics are captured on the worker node network interface cards. The first performance variable measures the average number of all network packet reads per second (APRps). The second variable captures the average number of all network packet writes per second (APWps). The measure "APIORkBs" refers to the amount of network I/O read traffic in kB per second sent between the servers. The fourth metric, "APIOWkBs," indicates the amount of network I/O write traffic in kB per second sent between the servers.

Fig. 3 illustrates the average network I/O (KB/s) on each Apache Spark node in tests 1-3 while Fig. 4 demonstrate the average network I/O (KB/s) on each Apache Spark node in tests 3-6.

In the perimeter-based network architecture, the average APRps reads per second are 637 across all Spark worker nodes. The average APWps writes per second are 620. The average APIORkBs read traffic between all Spark worker nodes is 80 while APIOWkBs is 78. The authors reconfigured the network architecture in the subsequent test to provide further insight into IDPS placement impact on distributed big data systems.

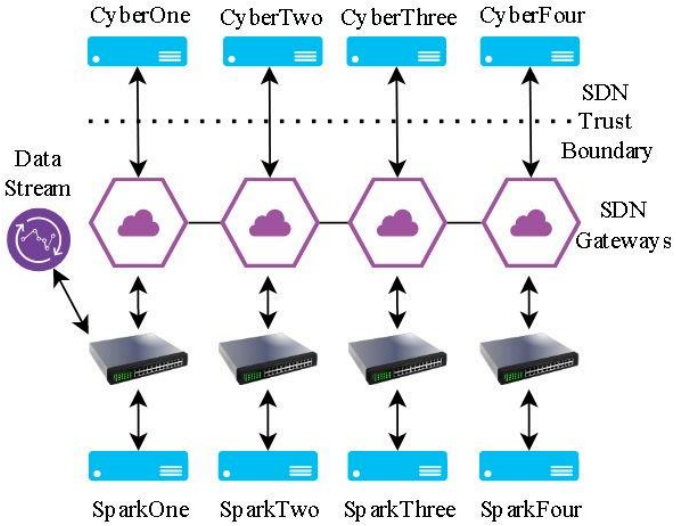

Fig. 2. Perimeter-less security network architecture.

B. Tests 2-3 Perimeter-less Security Results

Fig. 2 demonstrates the big data network designed for tests two and three. Network architecture uses a modified perimeter-less design proposed by Kotantoulas [39]. In contrast to the traditional perimeter IDPS location in Fig. 1, every big data worker node is in a zero trust network. The authors designed an SD-WAN trust boundary to secure each big data node. The boundary consists of Snort and Suricata intrusion detection and prevention security gateways. Similar to the virtual software defined perimeter (vEPC) proposed by Bello et al. [40], this study's zero trust software-based system acts as a security gateway for all distributed servers. Sparkone through Sparkfour are designed to operate securely in most cloud architectures in this model by integrating an SDN security stack on each physically distanced server. The integrated IDPS gateways control and authorize incoming and outgoing network communication. The design emulates the trust boundary surrounding the cloud edge in [39] using the SSH and TLS protocols. Gateways authenticate and connect the distributed systems using a 3072-bit key generated by the Rivest–Shamir–Adleman (RSA) algorithm.

Benchmark results for test 2 with Snort SDN gateways show the wait% is 0.413% and CPU% is 12.54%. Results from this study show that CPU resource consumption is over two times greater in the zero trust architecture than the perimeter network design. Test 3 with Suricata SDN gateways results in 11.05% CPU% and 0.342% wait%. Similar to the perimeter-less design in test 2, test 3 used considerably more CPU resources than test 1. Despite similar rulesets, Suricata SDN gateways used slightly less CPU than Snort.

In the test 2 perimeter-less network architecture the average APRps reads per second are 2,198 across all Spark worker nodes. The average APWps writes per second are 653. The average APIORkBs read traffic between all Spark worker nodes is 298 in test 2, APIOWkBs is 82.

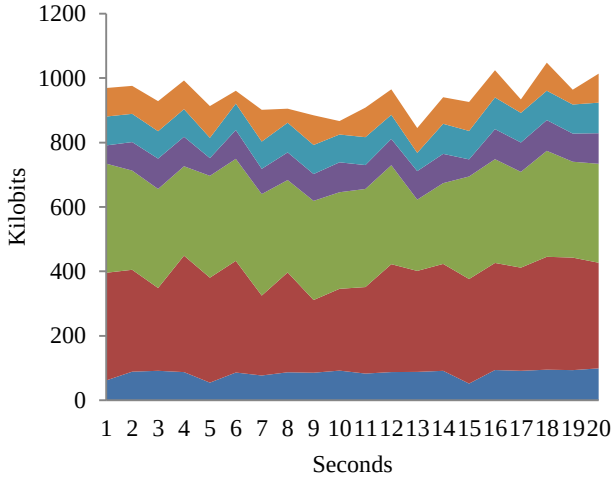

Fig. 3. Tests 1-3 spark per node network I/O in KB/s.

The test 3 network architecture had similar results to test 2. The average APRps reads per second are 2,120 across the distributed Spark systems. The average APWps is 611. APIORkBs between the big data servers is 289 and APIOWkBs is 77. Fig. 3 illustrates the average network I/O (KB/s) on each Apache Spark node in tests 1-3. These results indicate that network traffic and network I/O are nominal when writing to HDFS in all network architectures within this study. In contrast, the number of packets the systems have to read is higher in the perimeter-less network architectures. APRps is over three times higher in tests 2 and 3 than in test 1.

C. Test 4 Perimeter-Based DDoS Attack Results

Test 4 uses the network architecture (Fig. 1), parallel to test 1. Perimeter-based intrusion detection and prevention systems protect the internal LANs of the Spark nodes. CyberOne through CyberFour are active in test 4. The cyber servers are configured to flood the big data cluster with unlimited TCP SYN handshakes.

Benchmark results for the big data servers during the DDoS attacks parallel test 1 in test 4. In test 4, the IDPSs prevented additional CPU load and network load on the big data servers. In the test case, the hardware IPSs successfully blocked the DDoS attacks.

D. Tests 5-6 Perimeter-less DDoS Attack Results

Tests 5 and 6 are similar to tests 3 and 4. However, DDoS attacks are administered on the big data cluster. Tests 5-6 use the (Fig. 2) perimeter-less security network architecture. Test 5 uses the Snort-based SDN security boundary, while test 6 uses Suricata. CyberOne through CyberFour are active in tests 5 and 6. The cyber servers execute DDoS attacks on the big data cluster by flooding the servers with unlimited TCP SYN handshakes.

Snort and Suricata security gateways successfully protect the big data systems from DDoS attacks in a zero trust network in tests 5 and 6; however, at the expense of local computational resource increases. Results for test 5 with Snort SDN gateways show the wait% is 0.308% and CPU% is 13.8%. CPU resource consumption increases on average over 1% on the big data servers during the DDoS attacks. Test 6 with Suricata SDN gateways results in 11.95% CPU% and 0.337% wait%. DDoS attacks increased average CPU% by 0.9% across big data systems. Suricata SDN gateways used slightly less CPU than Snort SDN gateways during the DDoS attacks.

Within the test 5 perimeter-less network architecture the average APRps reads per second are 4,762 across all distributed by data secondary nodes. The average APWps writes per second are 626. The average APIORkBs traffic between the distributed systems is 425. APIOWkBs is 79.

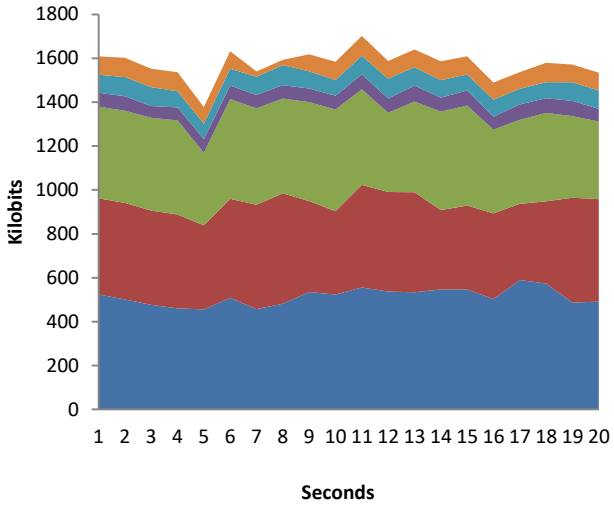

Fig. 4. Tests 4-6 spark per node network I/O in KB/s.

The Suricata gateways in test 6 have average APRps reads per second of 4,311 across the distributed Spark systems. Average APWps is 661. APIORkBs between the big data servers is 416 and APIOWkBs is 81. Fig. 4 demonstrates the average network I/O (KB/s) on each Apache Spark node in tests 3-6.

E. Test 7 Perimeter-Based DDoS Attack Results

Test 7 shares the same network architecture as test 1 and test 4, illustrated in Fig. 1. To decipher how the DDoS attacks affect the big data servers in the perimeter-based network architecture without IDPS protection, test 7 repeats test 4 but allow all network traffic from CyberOne through CyberFour to the big data cluster. When the DDoS attacks are allowed through the perimeter IPSs in the Fig. 1 network architecture, results show an average CPU% of 17.9% across all distributed big data systems. Predictably, network packets increase in test 7 compared to tests 1 and 4. APRps is 2,895 while APIORkBs is 518. Test 7 has the highest APIORkBs of all network benchmarks performed in this study.

F. Discussion of the Results

The results illustrate that network traffic and network I/O have marginal differences when writing to HDFS in the network architectures studied. CPU resources and network traffic read by the operating systems increased in zero trust network architectures. The most substantial differences were between tests 4 and 5. During the DDoS attacks, the big data servers required more CPU resources in the perimeter-less security network architecture. In test 5, APIORkBs are considerably higher at 425 than test 4 at 80. This additional traffic is partly due to the SDN security boundaries necessary to protect the systems in a zero trust network environment.

Shifting compute resources closer to individual devices may be necessary as network security perimeters dissipate. However, zero trust architectures in the experimental environment reduced cluster performance. Therefore, additional research is beneficial to optimize the design of perimeter-less network environments.

G. Limitations

Several environmental factors limit the results. Site-to-site networks were on leased 200 Mbps connections. Future studies might consider leased lines capable of establishing more robust data streams to the distributed nodes. A subsequent restriction is the number of architectures and communication technologies tested. Similar to the architecture in [40], gateways allow for IP Security (IPsec) or Transport Layer Security (TLS) protocols. Future IDPS SDN gateways could add this layer of encryption in a software-defined security boundary between geo-distributed big data systems. The outlined limitations emphasize the need for future research to investigate more extensive network architectures and IDPS technologies for big data system security.

V. Conclusion

As the volume of data expands, organizations require big data systems to perform large-scale data analytics. One of several needs for these systems is effective intrusion detection and prevention strategies. This paper builds a review of the

literature on methods used to reduce cybersecurity threats in a range of network architectures that big data systems operate. Findings from literature suggest intrusion detection and prevention systems can respond to certain security attacks. However, a potential disadvantage of capable security systems is the impact on big data system cluster performance. Using a design science approach, the authors develop an eight-step process to benchmark big data systems in varying network architectural environments. The new benchmark process is tested on real-time big data systems running in perimeter-based and perimeter-less network environments. During DDoS cyber-attacks, perimeter-based network architectures outperformed perimeter-less network architectures. This underlines the importance of optimizing the design of zero trust architectures for distributed big data systems.

REFERENCES

[1] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1717–1731, Jun. 2021, doi: 10.1109/JSYST.2020.2992966.

[2] I. D. Aiyanyo, S. Hamman, and H. Lim, "A systematic review of defensive and offensive cybersecurity with machine learning," Applied Sciences, vol. 10, no. 17, p. 5811, 2020, doi: 10.3390/app10175811.

[3] N. V. Patil, C. Rama Krishna, and K. Kumar, "Distributed frameworks for detecting distributed denial of service attacks: A comprehensive review, challenges and future directions," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 10, pp. 1-21, May 2021, doi: 10.1002/cpe.6197.

[4] R. Rafiq, M. J. Awan, A. Yasin, H. Nobanee, A. M. Zain, and S. A. Bahaj, "Privacy prevention of big data applications: A systematic literature review," *Sage Open*, vol. 12, no. 2, Apr. 2022, doi: 10.1177/21582440221096445.

[5] R. Mitchell and I. R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surv.*, vol. 46, no. 4, Mar. 2014, doi: 10.1145/2542049.

[6] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, Apr. 2017, doi: 10.1016/j.jnca.2017.02.009.

[7] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "Security-by-design in multi-cloud applications: An optimization approach," *Information Sciences*, vol. 454–455, pp. 344–362, Jul. 2018, doi: 10.1016/j.ins.2018.04.081.

[8] R. Atat, L. Liu, J. Wu, G. Li, C. Ye, and Y. Yang, "Big data meet cyber-physical systems: a panoramic survey," *IEEE Access*, vol. 6, pp. 73603–73636, 2018, doi: 10.1109/ACCESS.2018.2878681.

[9] R. Gifty, R. Bharathi, and P. Krishnakumar, "Privacy and security of big data in cyber physical systems using Weibull distribution-based intrusion detection," *Neural Computing and Applications*, vol. 31, no. 1, pp. 23–34, Jan. 2019, doi: 10.1007/s00521-018-3635-6.

[10] S. F. Ochoa, G. Fortino, and G. Di Fatta, "Cyber-physical systems, internet of things and big data," *Future Generation Computer Systems*, vol. 75, pp. 82–84, Oct. 2017, doi: 10.1016/j.future.2017.05.040.

[11] K. A. P. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque, "Internet of Things: A survey on machine learning-based intrusion detection approaches," *Computer Networks*, vol. 151, pp. 147–157, Mar. 2019, doi: 10.1016/j.comnet.2019.01.023.

[12] N. Moustafa, B. Turnbull, and K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019, doi: 10.1109/JIOT.2018.2871719.

[13] A. Yang, Y. Zhuansun, C. Liu, J. Li, and C. Zhang, "Design of intrusion detection system for Internet of Things based on improved BP neural network," *IEEE Access*, vol. 7, pp. 106043–106052, 2019, doi: 10.1109/ACCESS.2019.2929919.

[14] Z. Tan *et al.*, "Enhancing big data security with collaborative intrusion detection," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 27–33, Sep. 2014, doi: 10.1109/MCC.2014.53.

[15] A. N. Jaber and S. U. Rehman, "FCM–SVM based intrusion detection system for cloud computing environment," *Cluster Computing*, vol. 23, no. 4, pp. 3221–3231, Dec. 2020, doi: 10.1007/s10586-020-03082-6.

[16] M. Hafsa and F. Jemili, "Comparative study between big data analysis techniques in intrusion detection," *Big Data and Cognitive Computing*, vol. 3, no. 1, pp. 1-13, Dec. 2018, doi: 10.3390/bdcc3010001.

[17] F. M. Awaysheh, M. N. Aladwan, M. Alazab, S. Alawadi, J. C. Cabaleiro, and T. F. Pena, "Security by design for big data frameworks over cloud computing," *IEEE Transactions on Engineering Management*, pp. 1–18, Feb. 2021, doi: 10.1109/TEM.2020.3045661.

[18] A. Bocci, S. Forti, G. L. Ferrari, and A. Brogi, "Secure FaaS orchestration in the fog: How far are we?" Computing, vol. 103, no. 5, pp. 1025–1056, May 2021, doi: 10.1007/s00607-021-00924-y.

[19] Y. Wang, X. Zhang, Y. Wu, and Y. Shen, "Enhancing leakage prevention for mapreduce," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1558–1572, 2022, doi: 10.1109/TIFS.2022.3166641.

[20] O. Ohrimenko, M. Costa, C. Fournet, C. Gkantsidis, M. Kohlweiss, and D. Sharma, "Observing and preventing leakage in MapReduce," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2015, pp. 1570–1581. doi: 10.1145/2810103.2813695.

[21] A. M. Sauber, A. Awad, A. F. Shawish, and P. M. El-Kafrawy, "A novel hadoop security model for addressing malicious collusive workers," *Computational Intelligence and Neuroscience*, vol. 2021, pp. 1-10, 2021, doi: 10.1155/2021/5753948.

[22] P. Derbeko, S. Dolev, E. Gudes, and S. Sharma, "Security and privacy aspects in MapReduce on clouds: A survey," *Computer Science Review*, vol. 20, pp. 1–28, May 2016, doi: 10.1016/j.cosrev.2016.05.001.

[23] R. Poddar, T. Boelter, and R. Popa, "Arx: An encrypted database using semantically secure encryption," *Proceedings of the VLDB Endowment*, vol. 12, pp. 1664–1678, Jul. 2019, doi: 10.14778/3342263.3342641.

[24] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3369–3388, Fourthquarter 2018, doi: 10.1109/COMST.2018.2854724.

[25] S. T. March and G. F. Smith, "Design and natural science research on information technology," *Decision Support Systems*, vol. 15, no. 4, pp. 251–266, Dec. 1995, doi: 10.1016/0167-9236(94)00041-2.

[26] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004, doi: 10.2307/25148625.

[27] "Dell technology," *Dell Inc*, June, 2022. [Online]. Available: https://www.dell.com.

[28] "Cisco routers and SD-WAN," *Cisco Systems*, June, 2022. [Online]. Available: https://www.cisco.com/site/us/en/products/networking/sdwan-routers/index.html.

[29] "Benchmarking & Diagnostic Software," *Passmark Software*, June, 2022. [Online]. Available: https://www.passmark.com.

[30] "Spark tuning guide on 3rd generation Intel® Xeon® scalable processors based platform," *Intel Corporation*, August, 2021, [Online]. Available: https://www.intel.cn/content/www/cn/zh/developer/articles/guide/spark-tuning-guide-on-xeon-based-systems.html.

[31] "Tuning Spark," *The Apache Software Foundation*, July, 2022. [Online]. Available: https://spark.apache.org/docs/3.2.2/.

[32] "Cluster Mode Overview," *The Apache Software Foundation*, June, 2022. [Online]. Available: https://spark.apache.org/docs/latest/cluster-overview.html.

[33] "Apache Hadoop YARN," *The Apache Software Foundation*, June, 2022. [Online]. Available:

https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html.

[34] "Kali linux features," *OffSec Services Limited*, June, 2022. [Online]. Available: https://www.kali.org/features.

[35] "Snort FAQ/Wiki," *Cisco Systems*, July, 2022. [Online]. Available: https://www.snort.org/faq.

[36] "Suricata user guide," *Open Information Security Foundation*, July, 2022. [Online]. Available: https://suricata.readthedocs.io/en/suricata-6.0.6.

[37] "nmon for Linux," *IBM*, June, 2022. [Online]. Available: http://nmon.sourceforge.net.

[38] "Collapsed data center and campus core deployment guide," *Cisco Systems*, June, 2022. [Online]. Available: https://www.cisco.com/c/dam/global/en_ca/solutions/strategy/docs/sbaGov_nexus7000Dguide_new.pdf.

[39] J. Kotantoulas, "Zero trust for government networks," *Cisco Systems*, June, 2022. [Online]. Available: https://blogs.cisco.com/government/zero-trust-for-government-networks-6-steps-you-need-to-know.

[40] Y. Bello, A. R. Hussein, M. Ulema, and J. Koilpillai, "On sustained zero trust conceptualization security for mobile core networks in 5G and beyond," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1876–1889, Jun. 2022, doi: 10.1109/TNSM.2022.3157248.