Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Adaptive Intrusion Detection in Cloud Environments Using Change Point Analysis and Unsupervised Feature Monitoring

Abdulaziz Aldribi
Department of Computer Science, College of Computer, Qassim University, Buraydah, Saudi Arabia
aaldribi@qu.edu.sa

ARTICLE INFO

Received: 17 Dec 2024

Revised: 20 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

Securing virtualized infrastructures is a critical challenge in cloud computing due to dynamic resource allocation and sophisticated cyberattacks. Traditional intrusion detection systems (IDS) often fall short in addressing cloud-specific requirements such as scalability, elasticity, and diverse attack vectors. This work introduces CloudIDS, an intrusion detection system tailored for cloud environments. CloudIDS employs Principal Component Analysis (PCA) to extract key features from network traffic and applies Change Point Models (CPMs), including Mann-Whitney and Cramer-von-Mises statistics, to detect abrupt shifts in network behavior indicative of attacks.

Two Riemannian-based sliding window algorithms—chunking and rolling—enable the detection of stable and transient patterns in virtual machine (VM) traffic. Experiments using the ISOT-CID dataset, which covers various attack types (e.g., scanning, dictionary attacks, reconnaissance, and denial-of-service), demonstrate that CloudIDS achieves high detection accuracy with minimal delay compared to conventional methods. Parameter tuning, particularly Average Run Length (ARL0) and startup length, reveals trade-offs between detection speed and false positives. An ablation study further validates the critical roles of PCA feature extraction and Riemann-based windowing.

CloudIDS presents a flexible, adaptive solution for intrusion detection in cloud environments. Future work will focus on integrating real-time monitoring, reinforcement learning-based adaptation, and contextual metadata to further enhance detection performance accuracy.

Keywords: Cloud Computing, Intrusion Detection System, Network Security, Security, Anomaly Detection

Introduction

Cloud computing has changed how organizations provide, scale, and pay for compute and storage, but its multi-tenant, elastic fabric also expands the attack surface. Traffic volumes are orders of magnitude greater than in conventional data centers, virtual machines (VMs) can be spun up and down in a matter of seconds, and security responsibility is shared between cloud providers and tenants [1]. In this environment, Intrusion Detection Systems (IDS) continue to be an essential line of defense, tasked with identifying policy breaches, malware, and zero-day attacks in real time. Even after decades of IDS research, the majority of signature bases and learning pipelines were built for static, single-tenant networks. Simply transplanting them to the cloud tends to produce unaffordable false positives, east-west traffic blind spots, and virtual switch performance bottlenecks [2]. As a result, the community still does not have an integrated framework that (a) models cloud-specific attack vectors like VM-escape

or hypervisor tampering, (b) can scale with burst workloads, and (c) minimizes overhead. The absence of the integrated framework motivates this research.

There are three detection methods, misuse, anomaly, and hybrid, are complimentary philosophies of detecting intrusions:

- Misuse (Signature Based) Detection: Fast and precise for known malware but blind to zero days or polymorphic variants without constant updates—problematic in dynamic cloud workloads [3].
- Anomaly-Based Detection: Learns “normal” behavior and flags deviations, catching novel or insider attacks, but elastic cloud scaling can appear malicious, causing false alarms unless baselines adapt [4], [5].
- Hybrid Detection: Chain's signature filtering with anomaly analytics to boost both precision and recall; effective but requires careful threshold tuning and resource management to avoid excessive overhead on multi-tenant hosts [6].

The where of intrusion detection is as consequential as the how, because the vantage point dictates what telemetry an IDS can observe and how much overhead it imposes.

- Host based IDS (HIDS) reside within every VM's OS, providing rich, syscall level insight into file access, privilege raises, or kernel hooks; such fine-grained granularity is optimal for detecting insider attacks or rootkits, but the sensor fights with the tenant's CPU cycles and can be disabled itself if the VM has been compromised [7].
- Network-based IDS (NIDS) relocate the sensor to virtual or physical firewalls, switches, or cloud gateways, monitoring packet flows across several tenants. This wide span renders NIDS highly appropriate for volumetric attack detection, worm spreading, and east-west movement but threatens to suffocate deep packet inspection engines with encrypted traffic and raw throughput unless offloaded to smart NICs or based on flow-level metadata [8].
- Hypervisor-based IDS is deeply integrated into the virtual machine monitor (such as Xen's Domo or KVM's host kernel modules), providing privileged views of inter-VM traffic and low-level activity without traversing guest OSes. From this "golden" vantage point, they maintain tenant isolation without being easily tampered with, but they have to be surgically light in order not to balloon VM scheduling latency and have to be hardened against hypervisor escape exploits themselves [9].
- VM based IDS create a single sensor VM that consumes mirrored traffic or logs from adjacent instances; this approach makes lifecycle management easier—operators can patch or scale the sensor without handling production VMs—but relies on proper traffic mirroring and adds additional bandwidth overhead, possibly dropping intra host packets if virtual taps are not properly configured. Practically, cloud vendors tend to interweave a multi-layer fabric, hypervisor monitors for low overhead baselining, selective host agents for high value workloads, and edge NIDS for coarse traffic shaping—to strike a balance between coverage, performance, and operational complexity [10], [11].

When these models of IDS deployments are being replanted in active cloud environments, four real-world challenges continue to resurface. First, the sheer volume and speed of virtualized networks, hypervisors processing millions of packets per second, render the possibility of capturing all traffic retrospectively impossible, and sensors have to examine streams in real-time. Second, elastic topology fluctuations like auto scaling groups, serverless functions, and transient containers make legitimate traffic baselines fluctuate erratically, bewildering anomaly detectors that assume steady "normal" profiles. Third, resource competition is a real risk: an IDS that commandeers CPU, RAM, or I/O

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

bandwidth eliminates the very performance flexibility and cost savings cloud tenants are seeking. Lastly, data privacy and multi-tenancy limit deep packet inspection; sensors have to respect isolation borders and regulatory requirements, which might hinder payload visibility and make it more difficult for cross-tenant threat hunting.

This paper presents Cloud-based Intrusion Detection System (CloudIDS), an unsupervised, online intrusion detection system that operates within the hypervisor and identifies abrupt behavior changes in high-speed packet streams through sequential change point analysis. The system isolates cloud-unique traffic features, removes redundancy with principal component analysis, and subsequently employs light-weight, monitors without labelled data or deep payload inspection. Proven on the public ISOT CID dataset, CloudIDS detects scanning, dictionary, and insider attacks in minutes and does so with minimal overhead on hypervisor resources.

The contributions of this research are focused on the design and development of CloudIDS, an online, unsupervised cloud-based hypervisor-level intrusion detection system. CloudIDS utilizes sequential change point detection methods to detect abrupt changes in cloud network behavior without requiring labeled data or deep packet inspection. It proposes cloud-specific feature extraction with Principal Component Analysis (PCA) to reduce feature redundancy and facilitate real-time detection with low resource overhead. In contrast to conventional batch analysis methods, CloudIDS employs non-parametric sequential change point models, including the CPM framework, to facilitate real-time anomaly detection in high-speed network streams. The system integrates effective windowing techniques, such as Riemann chunking and rolling windows, to optimize detection accuracy, latency, and computational performance, thereby being appropriate for dynamic, large-scale cloud environments. Deployed at the hypervisor level without touching guest VMs, CloudIDS provides end-to-end visibility into internal and external traffic and preserves tenant isolation and privacy. Its light-weight design imposes zero CPU and memory overhead, making it possible to deploy even in high-performance cloud environments (e.g., 40 Gbps environments). The system was empirically evaluated on the ISOT-CID public cloud intrusion dataset with fast and accurate detection of various attacks within sub-minute latency. Lastly, a prototype deployment in a production OpenStack cluster at Compute Canada West validated the real-world practicality and efficacy of CloudIDS in multi-tenant cloud environments.

The remainder of the paper is structured as follows: Section 2 is a discussion of the state-of-the-artwork in the area of intrusion detection systems for cloud environments. Section 3 is a description of the different change point detection algorithms used in the CloudIDS framework. Section 4 gives an overview of the overall framework of cloud-based intrusion detection. Section 5 discusses the feature extraction process from cloud network traffic, while Section 6 presents the computation of such features. Section 7 gives information regarding the implementation of the presented framework. Section 8 reports the experimental results and analyzes the performance of the system for different attack scenarios. Section 9 is reserved for the ablation study, analyzing the effect of various components and parameters on the performance of the system. Section 10 concludes the paper and provides directions for future research in the area.

Related Work

Cloud-based Intrusion Detection Systems (IDS) are now critical to protecting cloud computing environments, where dynamic resource allotment and multi-tenant architecture impose new threats. Signature-based and anomaly-based methods of IDS that have been used in traditional settings are too often inadequate to identify continually changing and sophisticated threats in cloud environments. This has resulted in increasing interest in combining Machine Learning (ML) and Deep Learning (DL) methods to improve the adaptability and precision of IDS in cloud computing. Recent research has investigated the use of ML and DL for intrusion detection with datasets such as KDDCup99, suggesting classification, clustering, and hybrid models to overcome issues such as high false positive rates and poor detection of zero-day attacks. This survey emphasizes major breakthroughs in cloud-based IDS, their

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

shortcomings, and provides the ground for creating stronger, scalable, and smarter intrusion detection systems for cloud infrastructure.

Intrusion Detection Systems (IDS) are crucial to detect unauthorized behavior from outside. IDS is crucial to detect numerous types of attacks, defining intrusion detection as a classification problem. Typical categories include Denial of Service (DoS), probe, User to Root (U2R), Remote to Local (R2L), and normal traffic. IDS mechanisms typically fall into two broad categories: signature-based and anomaly-based [12]. Signature-based methods are efficient in discovering known attacks and generally have a low false positive rate. But they become ineffective in identifying new or zero-day attacks because they are based on pre-known patterns. They need frequent updates, so they are time-consuming to maintain. Anomaly-based IDS, on the other hand, can detect known and unknown threats, such as zero-day exploits [13]. Even with this benefit, they tend to have extremely high false positive rates and are sensitive to tuning. Intelligent and adaptive solutions are needed in the face of mounting numbers and sophistication of cyberattacks. To avoid these issues, we suggest the application of machine learning-based classification as well as clustering techniques. Classification models are trained on completely labeled sets of data, whereas clustering algorithms can identify patterns in unlabeled data without needing the data to be labeled first.

Over the last few years, a lot of research has gone into the application of machine learning methods to Intrusion Detection Systems (IDS). In [14], Support Vector Machines (SVMs) have been used to identify anomalies in the KDD dataset. Also, in [15], deep learning-based artificial neural networks have been proposed to build IDS models for anomaly detection on the same dataset. Their findings reported good detection accuracy and low false alarm rate, which surpassed many other existing techniques. The work in [16] proposed cascading classifiers to detect and classify outliers in the KDD dataset, even when the data was not uniformly distributed. In [17], the utilization of decision tree and random forest (RF) for anomaly detection was utilized, whereas [18] aimed to design a decision tree classifier for accurate intrusion detection and proved its efficacy through experimental evaluation on two datasets. This method exhibited better results in terms of Accuracy (ACC), Detection Rate (DR), and False Alarm Rate (FAR) compared to other approaches. Also, [19] suggested blending several machine learning algorithms into a hybrid framework, for which experimental results indicated that hybrid models perform better than single algorithms. Exhaustive surveys of machine learning-based IDS approaches are available in [20].

Some recent research studies have tried to improve upon the shortcomings of classic models. For example, [21] proposed a four-layer classification method to differentiate between four attack types in the KDD dataset with low overall and misclassification errors. In addition, they proposed reducing the feature size of the original dataset to improve accuracy and reduce computational complexity. Yet, mislabeled attack types were still not resolved. Conversely, [22] used multiple supervised, unsupervised, and outlier detection algorithms on the same data set with lower overall accuracy because attacks were misclassified. Anomaly detection and classification approaches based on machine learning have found widespread use with the KDD data set, with four differentiated types of attacks with considerably dissimilar traffic patterns. In [23], misclassification errors for classification techniques based on the KDD data set were low. However, such models might encounter problems when operating in dynamic multi-cloud settings, whose attack behaviors are more intricate and interwoven. In addition, the outdated KDDCup99 dataset would probably not capture existing network traffic and threats anymore [24]. As explained in [14], SVM has been utilized in data mining for predictive analytics with the use of the KDDCup99 IDS database for classification purposes based on neural networks. The model obtained 90% accuracy on the training set and 80% accuracy through 10-fold cross-validation on the test set. The literature discusses a wide range of classification and clustering methods, as well as unsupervised learning methods. The overall detection accuracy, however, falls short due to the misclassification of some attack types. We thus suggest the use of hybrid machine learning models to overcome these limitations and improve IDS performance.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

While every cloud service model possesses inherent benefits, each also faces specific challenges. Virtualization becomes a requirement for IaaS resource provisioning, but it has limitations that can diminish the long-run value of IaaS solutions [25]. Platform as a Service (PaaS) is encumbered with interoperability, host dependency, confidentiality, authorization, reliability, and scalability problems. By a similar token, Software as a Service (SaaS) will have to face security threats to authorization, authentication, data confidentiality, service dependability, and network surveillance [26]. Satisfying all these security considerations is important to cloud service providers [27][48][49].

The dynamic nature of the threat horizon means that malicious actors constantly improve their tools and methods to utilize vulnerabilities in the cloud environment [28]. The conventional Intrusion Detection Systems (IDS) tend to struggle when it comes to detecting changing trends in network packets. Therefore, researchers recommend using Machine Learning (ML) and Deep Learning (DL) techniques together to enhance IDS performance [29]. ML and DL play significant roles in various fields, such as finance, government, scientific studies, and cybersecurity [30]. Specifically, the ability of ML to cluster and classify data is crucial for improving cybersecurity applications [31].

Change Point Detection Algorithms In Cloudids

Change point detection simply wonders: Is the stream I am currently seeing still coming from the same statistical distribution as before? If not, the point in time when it changed is indicated as a change point. The area of study goes all the way back to 1950s quality control charts and is now found in genomics, finance, and, most importantly, network security monitoring [32]. Two general modes are present. Batch (offline) detection stores the entire sequence of NN observations and scans using likelihood ratio or Bayesian tests; this works well when the expected number of change points is small but is not feasible in clouds where traffic never subsides. Conversely, sequential (online) detection processes packets in sequence, maintaining a running statistic like CUSUM or EWMA and sending an alert as soon as a threshold is exceeded [33]. This formulation allows an algorithm to "reset" following an alert and keep on looking for additional shifts, and it's best suited for streams that may have numerous changes over time.

Why are Sequential Methods Mandatory for Clouds? A hypervisor might observe thousands of packets in an eye blink; storing full traces for offline analysis would fill disk and memory. Further, attackers can evolve mid-campaign, so the detector has to continue running after an initial alarm. These limitations make a case for algorithms that (i) consume unbounded streams, (ii) have constant-time updates, and (iii) tolerate repeated detections without human resets. The CloudIDS design, thus, takes sequential, non-parametric change point models.

CPM framework (R package) CloudIDS uses the Change Point Model (CPM) framework, which is a class of sequential hypothesis tests packaged in an R package. CPM considers a two-sample statistic at each conceivable split within the current window; the best statistic D_n is compared against a pre-calculated threshold. When D_n exceeds the threshold value, a change is reported at time t, and monitoring resumes from the next packet. CPM provides a selection of statistics, Mann-Whitney (location changes), Mood (scale), Lepage, Kolmogorov-Smirnov, Cramer von Mises, to name a few, both parametric and non-parametric. Since CloudIDS can't rely on normal traffic distributions, it uses the Mann-Whitney test by default, which operates with unknown, continuous distributions and still efficiently identifies several change points [34].

- Single point mode: With detectChangePoint(), CloudIDS can identify the initial meaningful deviation, appropriate for high assurance VMs that are quarantined immediately upon any anomaly. Experiments indicated that a startup window of 20 observations and an Average Run Length (ARL0) of 370 provided a balance between sensitivity and false alarms, identifying a dictionary attack within ~6 minutes of occurrence.
- Multi-point mode: With processStream(), the monitor resets itself after every alert, catching consecutive events like port scans followed by insider reconnaissance. Comparative testing

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

of five CPM statistics showed that Mann-Whitney and Cramer von Mises provided the lowest latency (≈1 minute) at reasonable ARL0 settings, particularly when used with a Riemann rolling window of 0.05 seconds.

Window management Schemes: Since traffic is ongoing, CloudIDS reduces packets to windows before passing them to CPM. Two schemes are employed:

- **Riemann Chunking**: Fixed-size windows (e.g., 2 s) provide homogeneous sample sizes but can smear short spikes.
- **Riemann Rolling**: Sliding window moves by an offset (0.05–1.5 s in experiments). Lower offsets make the system more responsive at the cost of correlated samples and increased rates of false alarms. Experimentally, an offset of 0.05 seconds with startup = 2000 and ARL0 = 500 best traded speed against accuracy for high volume hypervisor data.

Practical Outcomes: Operating on a single day of ISOT CID hypervisor traces, the sequential CPM pipeline, supplied with PCA reduced traffic features, identified scanning, dictionary, ping, and unsuccessful DoS attacks with sub-minute latency and minimal overhead. Change point estimates frequently preceded complete attack manifestation by several seconds, providing operators with an important reaction window.

Framework Of The Cloud Intrusion Detection

CloudIDS is designed as a hypervisor-resident, all-stream pipeline to translate live packet flows to security verdicts with second-level latency. The sensor resides on every virtual switch tap, thereby observing both north–south (tenant ↔ Internet) and east–west (VM ↔ VM) traffic without software installation in guests. Every packet is time-stamped with nanoseconds, parsed in RAM by a headless TShark process, and payloads discarded immediately to respect privacy and reduce I/O overhead. The produced header record rows go into an in-memory column store where vectorised operations compute dozens of statistics every few milliseconds. A stage involving incremental PCA retains only the four orthogonal components that explain > 90 % of variance, so memory is constant irrespective of traffic rate. They are the input to sequential, non-parametric change point monitors, which operate in $O_{(i)}$ per packet time. If a monitor's metric crosses its threshold, CloudIDS labels the shift with the offending VM, its severity, and the relevant features, then it can auto throttle or quarantine the instance through the cloud control plane. Because the pipeline is streaming end-to-end and keeps only sliding window state in memory, end-to-end alert latency stays in the low seconds even on 40 Gbps links, and RAM usage grows with window size, not traffic volume. Table I shows the key feature of the design choices in CloudIDS

Table I. Key Features and Design Choices In Cloudids

Features & References	Descriptions
Unstructured Nature of TCP Dumps [35]	Hypervisors capture traffic in PCAP format, which stores binary frames with no schema. CloudIDS relies on headless TShark filters to extract header fields, 5-tuple, flags, lengths, and timestamps, directly in memory. This avoids disk writes, minimizes parsing latency, and ensures the sensor operates at wire speed while respecting tenant confidentiality by dropping payloads.
Feature Space [36]	From each frame, CloudIDS derives 19 raw attributes and then applies automated relational transformations to yield higher-order metrics: flow frequencies, entropy of source/destination distributions, load ratios, and degree statistics for the VM-level traffic multigraph. Incremental PCA prunes redundancy, producing four principal features (f0–f3) that dominate variance and are cheap to update.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Characterization of Cloud Network Activities (Empirical) [37]	A day-long study on the ISOT-CID dataset showed that normal workloads yield stable feature distributions, whereas attacks trigger sharp shifts. Port-scan bursts increased identical DST-IP/DST-port frequency; dictionary attacks spiked entropy; insider file-transfer sessions altered load ratios. These observations justified choosing non-parametric change-point tests that react to full-distribution changes rather than mean shifts alone.

Cloud Network Feature Extraction

Feature engineering in CloudIDS happens in two concentric loops. On the outer loop, traffic from all the tap interfaces on a hypervisor is combined, creating a host-level multigraph whose edges tally packet counts and byte volumes between VM endpoints. From this graph, we calculate node degree, fan-out ratios, and hypervisor-wide entropy values—metrics that highlight macro events like DoS bursts or VM-escape chatter. The inner loop separates per-instance flows: each packet is assigned to its tenant VM and binned into rolling or chunked windows (Riemann/Lebesgue schemes). Within each window, CloudIDS tallies flow frequencies, computes inter-arrival jitter, and harvests directional statistics (inbound vs outbound) that are vital for detecting data exfiltration or internal scans.

Empirical examination of the ISOT-CID traces revealed that the entropy of destination ports drops in dictionary attacks, and load ratios (packet-rate ÷ peer count) peak prior to insider file transfers. To convert these disparate metrics into a sparse real-time vector, CloudIDS applies incremental PCA: every few milliseconds, the covariance matrix is recomputed, and only the four leading components, which account for over 90 % of variance, are passed to the sequential change-point detectors. Table II shows the feature groups employed by CloudIDS.

Table II. Feature Groups Used By Cloudids

Features & References	Description
Hypervisor-Oriented [38]	Builds a host-level multigraph of all VM-to-VM and VM-to-Internet flows, then extracts total byte/packet rates, node degrees, and cross-VM fan-out statistics to expose hypervisor-scale anomalies such as flooding or escape attempts.
Instance-Oriented [39]	It treats each VM as an independent sensor: it counts per-tuple frequencies, measures inbound/outbound asymmetry, and records inter-arrival jitter within rolling windows to reveal brute-force logins, lateral scans, or data-exfil bursts.
Entropy Features [40]	Computes Shannon entropy of source ports, destination IPs, and protocol mixes for both hypervisor and instance scopes; sharp drops indicate focused scans, while sudden rises suggest beaconing or botnet C2 rotation.
Load Features [41]	Derives load ratios—packet frequency divided by peer degree—and degree-based connection loads; abnormal surges flag impending DoS or heavy file-transfer sessions even before bandwidth peaks.
PCA [42], [43]	Runs incremental principal-component analysis to condense dozens of raw metrics into four orthogonal features (f0–f3) that preserve > 90 % variance, enabling constant-time updates for the change-point monitors.

Computing Features

After raw packet attributes are reduced to hypervisor- and instance-level measures, CloudIDS needs to choose when to pass those observations over to its change-point detectors. Two windowing techniques

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

are employed, one optimized for two different traffic behavior patterns seen within the ISOT-CID experience. Table III shows the two window techniques applied to average packet observations preceding feature computation and their detection latency implications.

Table III. Feature Groups Used By Cloudids

Features and References	Descriptions
Chunking Importance [44]	***Riemann chunking*** partitions the timeline into consecutive, non-overlapping windows of fixed length δt (e.g., 2 s). All packets falling inside a chunk are summarized once the window closes. Chunking offers statistical independence between windows—valuable for tests like Mann-Whitney—and yields deterministic memory usage, but may blur very short-lived spikes if the chunk is too large.
The Lebesgue Scheme for Computing Features [45]	***Lebesgue rolling*** shifts a window of N packets forward by an offset smaller than its length (e.g., 2000-packet window, 500-packet stride). This packet-count–based scheme aligns feature computation with actual traffic volume rather than wall-clock time, capturing micro-bursts that would be diluted in equal-time chunks. Rolling windows improve detection latency but introduce correlation between successive samples, demanding more conservative change-point thresholds.

Empirical tuning revealed a 2s Riemann chunk registered multi minute dictionary attacks with minimal false positives, and a 2000/500 Lebesgue rolling window revealed sub second port scan bursts in one minute. With support for both schemes, CloudIDS allows operators to choose a balance between independence (chunking) and responsiveness (rolling) that most suits their workload.

Framework Implementation

CloudIDS was prototyped within an in-production OpenStack cluster within Compute Canada West to ensure that design decisions hold up to actual packet rates and multi-tenant noise. The implementation takes a staged → streaming architecture:

1. **Packet Capture (Tap Layer):** A libpcap-based C-language sniffer is bound to each qbr tap interface that OpenStack's Neutron establishes for tenant vNICs. Packets are picked up in PROMISC mode, time-stamped by a NIC hardware clock, and duplicated into a lock-free ring buffer that ensures zero loss at a 40 Gbps line rate. Capture threads are pinned to dedicated CPU cores to prevent scheduler jitter and NUMA penalties.
2. **Header Extraction & Transport (ZeroMQ Layer):** Every capture thread removes payloads in situ, leaving behind only Ethernet, IP, TCP/UDP headers, and a 32-byte metadata trailer. The header records are encapsulated as 128-byte messages and pushed along a ZeroMQ pipeline socket to an analytics micro service colocated on the same hypervisor. This architecture avoids disk I/O and imposes tenant privacy by construction.
3. **In-Memory Columnar Store (Apache Arrow Layer):** The analytics service buffers 10k incoming messages and writes them out to an Arrow table with dictionary-encoded strings for IP addresses and ports, resulting in SIMD-friendly, cache-efficient storage. Windowing (Riemann chunking or Lebesgue rolling) is achieved by the Arrow record batch boundaries.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4. **Feature Engineering:** Vectorized C++ kernels calculate hypervisor-level graph metrics and VM-level frequency/entropy/load features within Arrow buffers directly. Outputs are added as new columns, without row materialization.
5. **Dimensionality Reduction:** An RcppEigen-based incremental PCA module pulls the feature matrix, refreshes the covariance estimate in O(k2) per batch (with k = 30), and outputs four principal components (f0–f3) explaining > 90 % of variance.
6. **Sequential Change-Point Detection:** The four elements flow into an embedded R interpreter in which CPM's processStream() keeps Mann-Whitney and Cramer–von Mises statistics. Thresholds are pre-tabulated for ARL0 = 500 to trade false alarms against latency. Upon a statistical firing, the service tags the responsible VM(s) and writes an alert to a Redis pub/sub channel.
7. **Mitigation Hooks:** A light Python listener takes in Redis notifications and, based on policy, calls OpenStack's Neutron API to rate limit the misbehaving port or Nova to suspend the VM. Notifications are also saved in Graylog for operator-level forensic analysis.

The whole stack resides in the host OS namespace (KVM) or Dom0 (Xen), i.e., no code executes within tenant VMs. On a CentOS 7 super node with 33 × 2.4 GHz cores and 248 GB RAM, maximum utilization under 40 Gbps bursts was 25 % CPU and 6 GB of memory, leaving plenty of headroom for regular cloud workloads.

ISOT Cloud Intrusion Dataset

To stringently test the introduced CloudIDS framework, we used the ISOT Cloud Intrusion Dataset (ISOT-CID)—a large-scale, publicly accessible dataset tailored for cloud-based intrusion detection studies. Derived from a production environment of a real-world OpenStack cloud, ISOT-CID is exceptional in terms of its scope, size, and variety, thus proving to be a precious resource for network behavior analysis in both normal and attack contexts. The ISOT-CID dataset was collected over the course of a few months and organized on several layers of the cloud stack, ranging from virtual machines (VMs) to hypervisors and network interfaces. There were two hypervisors and ten VM instances spread over three separate availability zones (Zones A, B, and C). The instances were subjected to several internal and external attacks mimicking real attacks and benign activities [46], [47].

The database holds extensive and annotated instances of various categories of attacks, including but not limited to:

- **External Attacks:** Denial of Service (DoS), masquerade, and remote login attacks.
- **Internal Attacks:** Privilege escalations, credential theft, botnet behavior, and data exfiltration.
- **Traffic Types:** Internal (hypervisor-to-hypervisor), local (VM-to-VM), and external (VM-to-outside).

All network traffic was seized with TCPdump in promiscuous mode and stored in regular packet capture (PCAP) format. Traffic was gathered both with and without payloads to suit different experimental requirements. Table IV presents an overview of the ISOT-CID dataset features.

Table IV. Feature Groups Used By Cloudids

Attribute	Details
Cloud Platform	OpenStack
Dataset Size	> 8 TB
Data Format	PCAP (with and without payload)
Collection Duration	Several months (VMs), multiple days (Network/Hypervisor)

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

No. of Hypervisors	2
No. of VM Instances	10 (across Zones A, B, and C)
Attack Scenarios	4 (each ~1 hour, across different days)
Attack Types	DoS, Masquerade, Dictionary, Trojan, Credential Theft, Botnet
Traffic Categories	External, Internal, Local
Data Layers	Network, Hypervisor, VM, CPU, Memory
Geographic Diversity	Attacks from Europe & North America
Labeling	Yes (Normal vs. Malicious)
Public Availability	Yes (via ISOT website with documentation)

The data is designed to accommodate batch and real-time detection settings, facilitating the conceptualization and testing of online detection systems like CloudIDS. It is specifically suited for anomaly detection models, unsupervised learning methods, and sequential change point detection because it has a high temporal resolution and size.

The diversity and density of ISOT-CID make it a suitable benchmark for cloud intrusion detection research. It enables a broad set of experimental configurations, such as feature extraction investigations, change detection methods, and performance scalability testing. Researchers are invited to examine the dataset with rich metadata and guidelines to aid reproducibility and useful comparisons.

Experimental Results And Discussion

To ensure the validity of the efficiency of the suggested CloudIDS framework, we performed a series of thorough experiments on a high-performance supercomputing machine (the specifications are presented in Table V. The assessment was performed using one complete day of hypervisor-level network traffic from the ISOT-CID dataset, namely corresponding to a single reported attack round (see Figure 1). The experimental scope was limited to the virtual machines installed in a single cloud node, and a detailed analysis was conducted on one target VM, which was named VM8 and participated in several attack scenarios within the chosen time frame.

Figure 1 shows an in-depth time sequence of network attacks carried out within a day on the ISOT-CID testbed, reflecting both internal and external threats within a real-time cloud scenario. The timeline starts with an external network scan attack at around 09:37 aimed at the cloud subnet. Following this, around 09:41, the attacker tried a dictionary attack against VM8 and managed to gain access. Upon entering, the attack was converted into an insider threat as the attacker performed some internal reconnaissance with a second network scan at 09:45. This was succeeded by malicious activities between 10:07 and 10:31, such as multiple login attempts and data transfers. At 10:30, the attacker launched a port scan on VM5, which resided on another cloud node, and then another dictionary attack at 10:36. Between 10:38 and 10:45, the system recorded several login attempts. The attack continued with a ping sweep at about 10:46 and file download at 10:47. The last stage consisted of an unsuccessful DoS attack on VM4 at 11:01 and ping to VM4 at 11:04. This chronology presents a good context to analyze the detection potential of CloudIDS and to demonstrate the escalation from external probing to large-scale internal exploitation. The chart highlights the attacker's shift from the outside to the inside, with striking multi-stage intrusion patterns in cloud systems.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

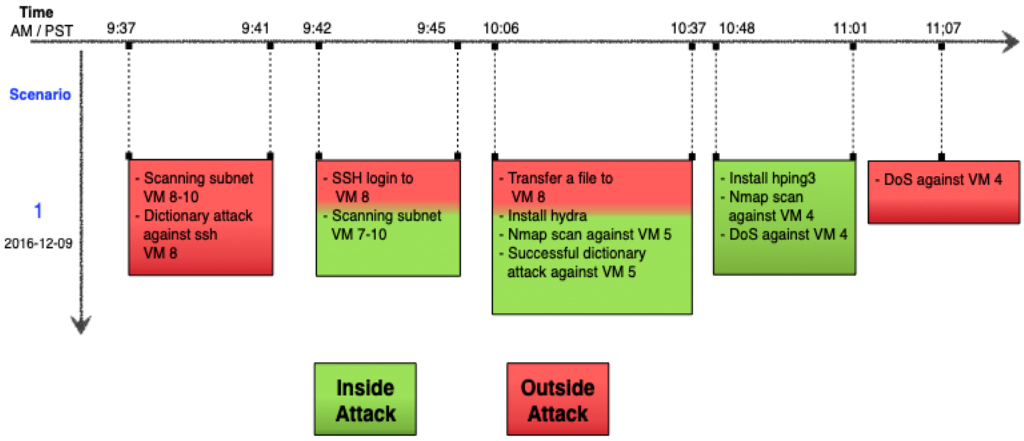


Fig.1. Attack Timeline

Table V. Supercomputing Machine Specification

OS	Specification
CentOS	RAM: 248 GB CPU: 33 cores Clock Speed: 2.40 GHz

A. OS Specification

To compare the performance of the proposed CloudIDS framework, experiments were performed on a high-performance CentOS-based system with 248 GB RAM, 33 CPU cores, and a clock speed of 2.40 GHz (see Table V). The goal was to detect change points in network traffic behavior as soon as possible and identify attack events correctly with minimal delay. The central analytic element of CloudIDS is the Change Point Model (CPM), utilized for single and multiple change point identification using univariate input, i.e., principal components obtained through PCA. As network traffic is non-Gaussian in nature, a variety of nonparametric CPM algorithms was used under sequential detection conditions. Feature extraction was preceded by PCA transformation to separate the most important variables contributing to behavioral changes. These characteristics were thereafter inspected via numerous CPM algorithms on all VM traffic, gathered at the hypervisor level. Two strategies for temporal segmentation were employed: a chunking method with a fixed offset of 2.0, and a rolling window method with varying offsets (1.5, 0.5, 0.1, and 0.05) to evaluate the effect of overlapping traffic windows on detection performance. More than 300 experiments were performed to optimize these parameters, with detailed results publicly made available in conjunction with the ISOT-CID dataset and its documentation.

B. Change Point Detection Using a Chunking Approach (CPM Package)

Single Point Detection In order to discover best configurations for single change point detection, we performed a number of experiments through the CPM (Change Point Model) package with emphasis placed on real-time anomaly detection from network traffic. Since no assumptions were made about the shape of the underlying data distribution, we utilized five nonparametric tests provided by the CPM package: Mann-Whitney (for location shifts), Mood (for scaling changes), and Lepage, Kolmogorov-Smirnov, and Cramer-von-Mises (for overall distributional changes). The detection function used in the CloudIDS framework is given by:

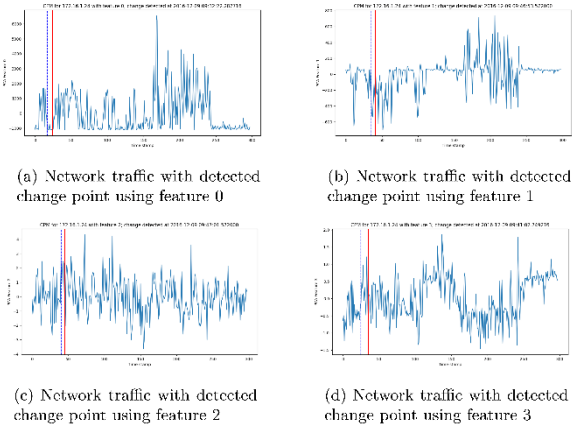
Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

detectChangePoint(pca_features_dst[:,feature], cpmType=cpmf, startup=st, ARL0=arl)

Here, **pca_features_dst[:, feature]** is the series of principal components from the PCA-transformed feature space, **cpmType** is the statistical test employed, **startup** is the number of startup observations needed before detection starts, and **ARL0** is the average run length, a false positive control threshold, which is computed from precomputed values included in the CPM package.

To measure the effect of these parameters, we tested different values for **startup** and **ARL0**. The best setting was found to be **startup = 20 and ARL0 = 370**, which gave the earliest and correct detection results. For example, employing the **Mann-Whitney** test on traffic data from VM8, the framework successfully detected a significant change point related to a dictionary attack as shown in Figure 2 . Feature f1 detected the change with an estimated change time of **09:41:02.740**, which was very close to the attack onset, and a detection time of **09:46:53.572**, representing a detection delay of about **5 minutes and 51 seconds**. This brief delay indicates the framework's capability for near real-time detection. Complete results for all VMs, employing various statistical tests, are listed in Table VI.



(a) Network traffic with detected change point using feature 0

(b) Network traffic with detected change point using feature 1

(c) Network traffic with detected change point using feature 2

(d) Network traffic with detected change point using feature 3

Fig.2. The plots shows the network traffic for VM8 with the estimated change point location as a blue dashed line and the detected change point as a red line. These change points found using Mann-Whitney CPM

Table VI. Single Change Point Detected Using Different Cpms For All Vms Using Riemann Chunking Scheme With **Startup** Is **St=20** And **Arl0** Is **Arl=370**.

PCA Features		f_0		f_1		f_2		f_3	
VMs IPs	CPMs	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time
	Mann-Whitney	09:59:10.363643	10:03:53.702664	09:35:29.277884	09:42:36.785094	09:48:31.766983	09:53:11.508307	Nothing detected	
	Mood	09:33:37.812500	09:48:29.766983	11:29:33.967937	11:29:39.967937	10:02:21.940607	10:02:27.940607	10:51:28.735671	10:52:34.560396
172.16.1.20	Lepage	09:59:10.363643	10:03:55.702664	09:35:27.247778	09:42:38.785094	10:02:21.940607	10:02:27.940607	10:51:28.735671	10:52:34.560396
	Kolmogorov-Smirnov	09:59:10.363643	10:03:55.702664	09:38:27.247778	09:42:38.785094	10:02:21.940607	10:02:27.940607	10:51:28.735671	10:52:34.560396
	Cramer-von-Miscs	09:59:10.363643	10:03:55.702664	09:38:27.247778	09:42:38.785094	09:48:31.766983	09:53:11.508307	11:38:16.449412	11:50:56.095093
	Mann-Whitney	10:03:58.238936	10:15:00.736581	10:00:39.887452	10:16:18.338728	Nothing detected		10:19:17.478158	10:55:31.218543
	Mood	Nothing detected							
172.16.1.21	Lepage	09:38:26.596199	10:00:39.887452	11:32:50.814371	11:35:18.056776	11:32:50.814371	11:35:18.056776	09:36:55.939748	09:48:23.192788
	Kolmogorov-Smirnov	10:03:58.238936	10:25:02.581316	10:09:17.488226	10:16:18.338728	09:49:52.088012	09:53:11.109883	10:03:58.238936	10:09:17.488226
	Cramer-von-Miscs	10:03:58.238936	10:15:00.736581	Nothing detected				09:36:55.939748	09:49:52.088012
	Mann-Whitney	09:32:08.287716	09:32:22.287716	09:41:02.749276	09:46:53.572000	09:45:37.409254	09:47:01.572000	09:32:24.287716	09:41:02.749276
	Mood	10:36:54.212882	10:44:02.315555	10:49:54.597499	10:52:36.573784	09:32:26.287716	09:38:29.491698	10:06:41.286575	10:13:52.218338
172.16.1.24	Lepage	09:32:08.287716	09:32:24.287716	09:41:02.749276	09:48:21.381128	09:32:26.287716	09:35:20.388633	09:23:23.731569	09:50:10.137962
	Kolmogorov-Smirnov	09:32:32.287716	09:49:52.137962	09:36:55.713451	09:46:59.572000	09:45:37.409254	09:47:01.572000	09:32:26.287716	09:38:29.491698
	Cramer-von-Mises	09:32:08.287716	09:32:22.287716	09:41:02.749276	09:46:53.572000	09:45:37.409254	09:47:01.572000	09:32:24.287716	09:41:02.749276
	Mann-Whitney	09:53:15.138159	10:00:42.883326	09:41:04.755373	09:42:37.356294	09:38:32.861860	09:44:11.838361	09:35:31.174154	09:41:02.755373
	Mood	09:41:02.755373	09:48:31.672526	10:00:38.883326	10:02:19.776088	09:35:29.174154	09:38:26.861860	09:53:09.138159	09:54:38.024761
172.16.1.27	Lepage	09:59:07.408171	10:00:42.883326	09:41:04.755373	09:42:33.356294	09:35:31.174154	09:41:06.755373	09:38:32.861860	09:42:33.356294
	Kolmogorov-Smirnov	09:59:07.408171	10:00:42.883326	10:00:38.883326	10:02:23.776088	09:38:32.861860	09:42:31.356294	10:47:00.952238	10:49:52.695589
	Cramer-von-Mises	09:59:07.408171	10:00:42.883326	09:41:04.755373	09:42:37.356294	09:38:32.861860	09:42:33.356294	09:35:31.174154	09:41:06.755373

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

To assess the performance of our extracted features in enabling continuous change point monitoring, we examined their contribution to early detection and compared the performance of different CPM algorithms under different settings. Our main interest during this phase was network traffic related to virtual machines participating in the initial round of attacks.

Throughout the first attack window **(09:37–09:41)**, the attacker started by performing a subnet scanning activity against the IP addresses **172.16.1.21–27**, after which there was a dictionary attack against VM8. This activity of attack should materialize in terms of large departures in the pattern of network traffic. Since our system makes use of single change point detection, we sought to establish whether these changes were quickly detected, and which features were most revealing of the intrusion.

As detailed in Table VII, change points were detected effectively by at least two of the features that were extracted, with **f1** and **f3** being the most sensitive and effective in anomaly detection. Of the statistical tests employed, **Mood's test** was least responsive to this attack scenario. However, **Mann-Whitney, Lepage,** and **Cramer-von-Mises** statistics provided better performance, both in detection and latency. The mean detection delays and the corresponding estimated change point times of features **f1** and **f3** of these three CPM algorithms are presented in Table VIII and reveal the relative detection efficiencies of each of them.

Table VII. Change Point Detected By Each Feature (In Bold}) Which Was Caused By Attacker Activities

PCA Features		f_0		f_1		f_2		f_3	
VMs IPs	CPMs	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time
172.16.1.20	Mann-Whitney	09:59:10.363643	10:03:53.702664	**09:35:29.277884**	**09:42:36.785094**	09:48:31.766983	09:53:11.508307	Nothing detected	
	Mood	**09:33:37.812500**	**09:48:29.766983**	11:29:33.967937	11:29:39.967937	10:02:21.940607	10:02:27.940607	10:51:28.735671	10:52:34.560396
	Lepage	09:59:10.363643	10:03:55.702664	**09:35:27.247778**	**09:42:38.785094**	10:02:21.940607	10:02:27.940607	10:51:28.735671	10:52:34.560396
	Kolmogorov-Smirnov	09:59:10.363643	10:03:55.702664	**09:38:27.247778**	**09:42:38.785094**	10:02:21.940607	10:02:27.940607	10:51:28.735671	10:52:34.560396
	Cramer-von-Mises	09:59:10.363643	10:03:55.702664	**09:38:27.247778**	**09:42:38.785094**	09:48:31.766983	09:53:11.508307	11:38:16.449412	11:50:56.095093
172.16.1.21	Mann-Whitney	10:03:58.238936	10:15:00.736581	10:00:39.887452	10:16:18.338728	Nothing detected		10:19:17.478158	10:55:34.218543
	Mood	Nothing detected							
	Lepage	**09:38:26.596199**	**10:00:39.887452**	11:32:50.814371	11:35:18.056776	11:32:50.814371	11:35:18.056776	**09:36:55.939748**	**09:48:23.192788**
	Kolmogorov-Smirnov	10:03:58.238936	10:25:02.581316	10:09:17.488226	10:16:18.338728	09:49:52.088012	09:53:11.109883	10:03:58.238936	10:09:17.488226
	Cramer-von-Mises	10:03:58.238936	10:15:00.736581	Nothing detected				**09:36:55.939748**	**09:49:52.088012**
172.16.1.24	Mann-Whitney	09:32:08.287716	09:32:22.287716	**09:41:02.749276**	**09:46:53.572000**	09:45:37.409254	09:47:01.572000	**09:32:24.287716**	**09:41:02.749276**
	Mood	10:36:54.212882	10:44:02.315555	10:49:54.597499	10:52:36.573784	**09:32:26.287716**	**09:38:29.491698**	10:06:41.286575	10:13:52.218338
	Lepage	09:32:08.287716	09:32:24.287716	**09:41:02.749276**	**09:48:21.381128**	**09:32:26.287716**	**09:35:20.388633**	09:23:23.731569	09:50:10.137962
	Kolmogorov-Smirnov	**09:32:32.287716**	**09:49:52.137962**	**09:36:55.713451**	**09:46:59.572000**	09:45:37.409254	09:47:01.572000	**09:32:26.287716**	**09:38:29.491698**
	Cramer-von-Mises	09:32:08.287716	09:32:22.287716	**09:41:02.749276**	**09:46:53.572000**	09:45:37.409254	09:47:01.572000	**09:32:24.287716**	**09:41:02.749276**
172.16.1.27	Mann-Whitney	09:53:15.138159	10:00:42.883326	**09:41:04.755373**	**09:42:37.356294**	**09:38:32.861860**	**09:44:11.838361**	**09:35:31.174154**	**09:41:02.755373**
	Mood	**09:41:02.755373**	**09:48:31.672526**	10:00:38.883326	10:02:19.776088	**09:35:29.174154**	**09:38:26.861860**	09:53:09.138159	09:54:38.024761
	Lepage	09:59:07.408171	10:00:42.883326	**09:41:04.755373**	**09:42:33.356294**	**09:35:31.174154**	**09:41:06.755373**	**09:38:32.861860**	**09:42:33.356294**
	Kolmogorov-Smirnov	09:59:07.408171	10:00:42.883326	10:00:38.883326	10:02:23.776088	**09:38:32.861860**	**09:42:31.356294**	10:47:00.952238	10:49:52.695589
	Cramer-von-Mises	09:59:07.408171	10:00:42.883326	**09:41:04.755373**	**09:42:37.356294**	**09:38:32.861860**	**09:42:33.356294**	**09:35:31.174154**	**09:41:06.755373**

Table VIII. The Average Detection Delay And The Change Point Estimate Time Average Using F1 And F3 For A Single Change Point Detection.

	f_1		f_3	
	Average detection delay	Change point estimate time average	Average detection delay	Change point estimate time average
Mann-Whitney	7m 2s 571ms	2m 12s 261ms	4m 2s 752ms	-3m 2s 269ms
Lepage	7m 31s 174ms	2m 11s 584ms	10m 2s 229ms	-4m 2s 489ms
Cramer-von-Mises	7m 3s 238ms	3m 11s 584ms	7m 0s 531ms	-2m 2s 866ms

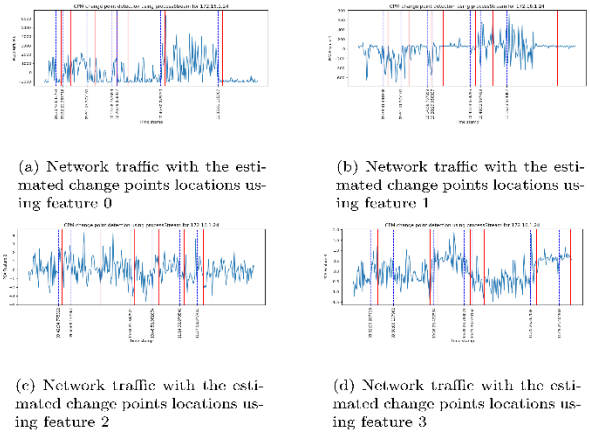
Generally, single change point detection is appropriate for high-security situations where even slight variations in network activity are deemed important. In these situations, administrators might prefer to prompt instant action, like isolating or completely shutting down a virtual machine, when one anomaly has been detected. This is a limiting application scenario, though. In more standard operational environments, where ongoing monitoring is necessary and the VM stays active even as behavior changes, there is a need to utilize detection mechanisms that can detect multiple change points across time. This allows for ongoing monitoring of changing threats and enables dynamic response plans consistent with real-world cloud operations.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Multiple Change Point Detection In the case of continuous monitoring, detecting multiple change points that could sequentially happen within network traffic data streams is important. To make this possible, we used the **processStream** function of the CPM package to extend the **detectChangePoint** functionality with the ability to automatically restart the detection after each detected change point. This makes it possible to continuously analyze new incoming observations without the need for manual intervention or batch processing.

In our experiments, the **processStream** procedure was used to identify multiple change points in sequences of network traffic readings, with the same parameter values used in the single change point identification experiments, namely a startup value of 20 and an average run length (ARL0) of 370 for each CPM statistical test.



(a) Network traffic with the estimated change points locations using feature 0

(b) Network traffic with the estimated change points locations using feature 1

(c) Network traffic with the estimated change points locations using feature 2

(d) Network traffic with the estimated change points locations using feature 3

Fig.3. The plots shows the network traffic for VM8 with the estimated change points locations as a blue dashed lines and the detected change points as a red lines. These change points found using Mann-Whitney CPM

Figure 3 shows the detection of multiple change points on all four main components derived from VM network traffic, with the Mann-Whitney statistic. Red lines are used to visualize detected change points, and blue dashed lines represent the corresponding estimated change times. After detecting a change, the CPM model is reset and continues to monitor from the next arriving observation, allowing real-time, adaptive detection.

A systematic comparison of all five CPM techniques, Mann-Whitney, Mood, Lepage, Kolmogorov-Smirnov, and Cramer-von-Mises, was performed using the Riemann chunking scheme. Detection results for VM8, under the first attack scenario, are presented in Tables IX and X.

Table IX. Multiple Change Points Detected In All Features Using Cpms For Vm8 With Riemann Chunking Scheme, With Startup=St20 And Arl0=Arl370.

PCA Features		f_0			f_1			f_2			f_3		
IP 172.16.1.24	Change point estimated time	Detection time	Thresholds	Change point estimated time	Detection time	Thresholds	Change point estimated time	Detection time	Thresholds	Change point estimated time	Detection time	Thresholds
Mann-Whitney	09:30:49.033764	09:32:18.287716	3.092	09:38:31.401608	09:44:04.888386	3.083	09:42:34.775312	09:46:57.572000	3.088	09:32:22.287716	09:38:31.401608	3.069
	09:32:22.287716	09:41:02.749276	3.085	09:51:41.277161	09:59:11.427766	3.101	09:50:06.137962	10:16:14.683406	3.121	09:50:06.137962	10:21:51.521227	3.111
	09:51:43.277161	10:00:38.895899	3.113	10:14:58.787659	10:19:06.912579	3.115	10:34:21.847925	10:36:48.212882	3.118	10:26:23.422594	10:29:13.782029	3.119
	10:14:58.787659	10:20:29.058037	3.117	10:20:29.058037	10:29:17.782029	3.124	10:46:58.962074	10:52:36.573784	3.127	10:39:23.783021	10:44:04.315555	3.126
	10:20:29.058037	10:29:15.782029	3.124	10:42:23.554795	10:44:06.315555	3.116	11:14:38.970548	11:17:11.360085	3.120	10:45:30.079150	10:57:01.877313	3.108
	10:45:30.079150	10:49:54.597499	3.115	10:49:52.597499	11:00:01.204410	3.117	11:27:58.077403	11:31:00.456259	3.108	11:35:26.362086	11:40:49.251150	3.319
	11:37:01.135707	11:40:47.251150	3.103	11:16:23.524867	11:46:54.136636	3.113				11:49:24.404969	11:54:28.156129	3.254
Mood	10:36:50.212882	10:43:38.315555	3.257	10:49:50.597499	10:49:58.597499	3.259	09:32:22.287716	09:36:35.713451	3.190	09:06:37.286575	10:13:48.718338	3.248
	10:46:58.962074	10:49:58.597499	3.259	11:14:38.970548	11:20:53.718977	3.258	09:49:54.137962	10:21:53.521227	3.252	10:13:54.218338	10:19:04.912579	3.264
				11:35:22.362086	11:40:49.251150	3.266	10:35:27.897294	10:36:48.212882	3.253			
				1:38:20.666103	11:43:20.914142	3.262	10:43:58.315555	11:45:54.442372	3.271			
Lepage	09:30:42.033764	09:32:20.287716	13.258	09:38:31.401608	09:46:35.572000	14.259	09:32:22.287716	09:32:28.287716	13.615	09:22:00.337624	09:50:06.137962	14.482
	09:32:20.287716	09:42:34.775312	14.109	09:51:43.277161	09:59:11.427766	14.630	09:32:28.287716	09:54:40.025467	14.550	10:21:47.521227	10:21:51.521227	14.767
	10:03:53.410851	10:06:33.286575	14.691	10:14:58.787659	10:19:06.912579	14.814	10:35:27.897294	10:36:18.212882	15.011	10:26:23.422594	10:31:46.806592	13.175
	10:17:45.871876	10:19:06.912579	14.814	10:20:29.058037	10:29:21.782029	14.881	10:46:58.962074	11:05:48.578760	15.097	10:42:27.554795	10:44:06.315555	17.100
	10:20:29.058037	10:29:17.782029	14.980	10:42:53.554795	10:49:52.597499	15.061	11:14:38.970548	11:20:53.718977	15.057	11:35:26.362086	11:40:49.251150	17.458
	10:46:58.962074	10:49:52.597499	15.061	11:14:38.970548	11:26:35.854973	15.060	11:31:00.456259	11:33:47.225457	15.064	11:40:53.251150	11:45:48.442372	13.824
	11:37:01.135707	11:40:47.251150	15.092	11:35:22.362086	11:41:55.273742	15.216	11:33:47.225457	11:44:21.553519	15.129	11:48:18.062110	11:49:24.404969	14.405
				11:37:01.135707	11:43:20.914142	15.183				11:49:24.404969	11:54:28.156129	13.295
Kolmogorov-Smirnov	09:32:22.287716	09:48:21.381128	0.998	09:38:31.401608	09:46:35.572000	0.998	09:42:34.775312	09:46:57.572000	0.998	09:32:22.287716	09:36:55.713451	0.998
	09:51:43.277161	09:59:11.427766	0.998	09:51:43.277161	09:59:09.427766	0.998	09:50:06.137962	10:06:13.286575	0.998	09:50:06.137962	10:13:50.218338	0.998
	10:17:45.871876	10:30:29.058037	0.998	10:26:23.422594	10:29:15.782029	0.998	10:32:57.457239	10:44:00.315555	0.998	10:26:23.422594	10:37:56.678494	0.998
	10:20:29.058037	10:31:46.806592	0.998	10:38:05.122231	10:44:04.315555	0.998	10:49:50.597499	10:52:36.573784	0.998	10:35:27.897294	10:39:48.212882	0.998
	10:46:58.962074	10:49:52.597499	0.998	10:49:52.597499	10:52:36.573784	0.998	11:14:38.970548	11:23:37.208906	0.998	10:42:25.554795	10:45:30.079150	0.998
	11:37:01.135707	11:40:49.251150	0.998	11:16:23.524867	11:27:58.077403	0.998	11:27:58.077403	11:29:31.937862	0.998	10:45:30.079150	11:05:18.578709	0.998
				11:35:24.362086	11:40:53.251150	0.998				11:14:12.970548	11:19:21.633716	0.998
										11:35:26.362086	11:40:49.251150	0.998
										11:40:53.251150	11:45:44.442372	0.998
Cramer-von-Mises	09:30:42.033764	09:32:18.287716	6.353	09:38:31.401608	09:44:04.888386	6.275	09:42:34.775312	09:46:57.572000	6.253	09:32:20.287716	09:38:31.401608	6.236
	09:32:22.287716	09:41:02.749276	6.233	09:51:43.277161	09:59:09.427766	6.256	09:50:06.137962	10:06:47.286575	6.291	09:50:06.137962	10:13:52.218338	6.322
	09:51:43.277161	09:59:09.427766	6.256	10:14:58.787659	10:19:06.912579	6.302	10:34:21.847925	10:36:48.212882	6.316	10:26:23.422594	10:29:13.782029	6.322
	10:14:58.787659	10:20:29.058037	6.279	10:20:29.058037	10:29:17.782029	6.354	10:49:50.597499	10:52:36.573784	6.337	10:39:23.783021	10:44:04.315555	6.336
	10:20:29.058037	10:29:17.782029	6.354	10:42:23.554795	10:44:04.315555	6.336	11:14:38.970548	11:17:11.360085	6.353	10:45:30.079150	10:55:36.025238	6.334
	10:46:58.962074	10:49:54.597499	6.306	10:49:52.597499	11:00:01.204410	6.320	11:27:58.077403	11:29:33.937862	6.312	11:14:42.970548	11:20:53.718977	6.305
	11:37:01.135707	11:40:45.251150	6.329	11:35:24.362086	11:45:54.442372	6.306				11:35:26.362086	11:40:53.251150	6.347
										11:40:53.251150	11:48:16.062110	6.312

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table X. Multiple Change Points Detected In All Features Using Cpms For Vm8 With Riemann Chunking Scheme, With Startup=St20 And Arl0=Arl370.

Attack periods		*09:37 - 9:41*	*09:42 - 9:45*	*10:06 - 10:37*	*10:43 - 10:48*	*10:48 - 11:07*
PCA Features	**CPMs**	Scanning, Dictionary attack	Scanning	Transfer a file, Download a Program, Scanning, Dictionary attack	Ping the same machine, Download a Program	Failed DoS attack, Ping VM4
f_0	Mann-Whitney	∼Yes (-5min)	No	Yes	Yes	No
	Mood	No	No	Yes	Yes	No
	Lepage	∼Yes (-5min)	No	Yes	Yes	No
	Kolmogorov-Smirnov	∼Yes (-5min)	No	Yes	Yes	No
	Cramer-von-Mises	∼Yes (-5min)	No	Yes	Yes	No
f_1	Mann-Whitney	Yes	No	Yes	Yes	Yes
	Mood	No	No	No	No	Yes
	Lepage	Yes	No	Yes	Yes	∼Yes (+7min)
	Kolmogorov-Smirnov	∼Yes (-5min)	No	Yes	Yes	Yes
	Cramer-von-Mises	Yes	No	Yes	Yes	Yes
f_2	Mann-Whitney	No	Yes	Yes	Yes	∼Yes (+7min)
	Mood	∼Yes (-5min)	No	Yes	Yes	No
	Lepage	∼Yes (-5min)	No	Yes	Yes	∼Yes (+7min)
	Kolmogorov-Smirnov	No	Yes	Yes	Yes	∼Yes (+7min)
	Cramer-von-Mises	No	Yes	Yes	No	∼Yes (+7min)
f_3	Mann-Whitney	Yes	∼Yes (+5min)	Yes	Yes	No
	Mood	No	No	Yes	No	No
	Lepage	No	No	Yes	Yes	No
	Kolmogorov-Smirnov	∼Yes (-5min)	∼Yes (+5min)	Yes	Yes	∼Yes (+7min)
	Cramer-von-Mises	Yes	∼Yes (+5min)	Yes	Yes	∼Yes (+7min)

The results show that the features extracted play an important role in the detection of changes in behavior, with each of them able to indicate network anomalies in at least three different phases of attack. On average, the estimated delay was about six minutes behind the real attack onset, which is tolerable in online detection scenarios. Among the tested methods, Mann-Whitney and Cramer-von-Mises performed best consistently in both detection accuracy and responsiveness. Consequently, further analysis is centered on these two CPM algorithms, tested further under the Riemann rolling scheme. The full set of experimental results, including results for all CPM variants, is provided to researchers together with the ISOT-CID dataset and its accompanying documentation.

C. Multiple Change Point Detection Using the CPM Package with Rolling Window

Since we are interested in detecting multiple change points, this section will present results for multiple change points detected by the CPM package. As mentioned before, we used different offsets for window size in the rolling scheme for CPM. In the following, the results will be discussed.

CPM Package For the CPMs in the rolling scheme, the analysis from the previous section is repeated, except we focus our experiments on two CPMs based on the previous results, namely, the Mann-Whitney statistic and Cramer-von-Mises.

Moreover, we use different values for startup and ARL0 with each cpmType. Through conducting extensive experiments, we noticed that, by choosing a small number of observations, the detection will be more sensitive to small changes and faster to detect change points. This will come at the cost of higher false positives. On the other hand, choosing ARL to be a large number will let the algorithm learn more and increase the detection accuracy, but with higher chances of false negatives, and the detection of the change points is delayed. This is clearly shown in our setting choice for offset 0.1. Therefore, the appropriate value of the number of observations depends on the activities being considered for the monitored VM. Based on the network traffic for VM 8, we applied these settings and performed experiments. The obtained results are discussed and presented next.

For the first offset 0.05 in rolling scheme, the optimum setting found regarding the startup is st=2000, and for ARL0 is arl=500. The detected multiple change points for both CPMs are presented in Fig. 4 and summarized in Table XI.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

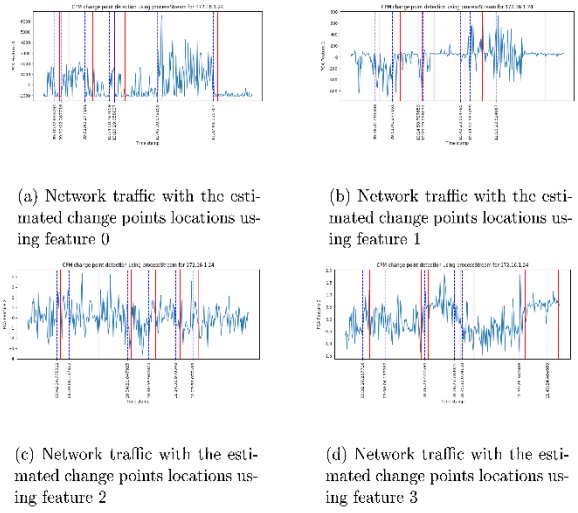
Research Article

Table XI. Multiple Change Points Detected Using Cpm For Vm8 With Riemann Rolling Scheme With Offset=0.05, Startup=St2000 And Arlo=Arl500.

PCA Features	f_0		f_1		f_2		f_3	
IP 172.16.1.24	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time
Mann-Whitney	09:45:37.409254	09:51:41.777161	09:38:31.541698	09:51:41.777161	09:42:35.725312	09:51:41.777161	09:32:20.487716	09:51:41.777161
	09:51:43.577161	10:21:47.621227	09:51:43.827161	10:17:44.971876	09:50:08.337962	10:20:29.158037	09:50:09.737962	10:11:01.591822
	10:13:55.918338	10:29:26.632029	10:27:54.378424	10:29:26.882029	10:26:25.022594	10:29:21.082029	10:26:25.372594	10:29:22.482029
	10:26:25.172594	10:41:00.168147	10:49:53.297499	10:49:58.347499	10:34:22.997925	10:49:56.197499	10:42:27.354795	10:49:56.547499
	10:44:03.765555	10:49:56.347499	11:11:52.471524	11:32:31.138038	10:48:23.402868	11:14:40.520548	10:49:57.797499	11:27:58.277403
	10:46:59.962074	11:29:30.937862	11:35:21.612086	11:41:57.273712	11:14:39.020548	11:29:38.237862	11:25:13.245442	11:35:15.012086
	10:49:57.847499	11:29:37.337862	11:45:48.792372	11:54:24.956129	11:37:02.935707	11:43:17.264142	11:35:26.812086	11:46:56.086533
	10:57:03.677313	11:35:15.062086						
	11:00:02.754410	11:35:23.962086						
	11:37:02.685707	11:40:45.601150						
Cramer-von-Mises	09:45:37.409254	09:51:41.777161	09:38:31.541698	09:51:41.777161	09:42:35.725312	09:51:41.777161	09:32:19.487716	09:51:41.777161
	09:51:43.527161	10:21:47.621227	09:51:43.827161	10:17:44.971876	09:50:08.137962	10:20:29.158037	09:50:09.737962	10:09:18.329662
	10:13:55.918338	10:29:22.482029	10:27:54.378424	10:29:20.882029	10:26:25.022594	10:29:20.882029	10:26:25.372594	10:29:22.482029
	10:26:25.172594	10:41:00.168147	10:38:06.073231	10:49:56.197499	10:34:22.997925	10:49:56.197499	10:42:27.704795	10:49:56.547499
	10:46:59.962074	10:49:56.347499	10:46:59.512074	11:25:10.595442	10:48:23.402868	11:14:40.520548	10:45:30.179150	11:27:58.627403
	10:49:57.797499	11:29:37.337862	11:14:39.220548	11:29:38.237862	11:14:39.020548	11:29:38.237862	10:49:57.797499	11:29:36.287862
	10:57:03.627313	11:35:15.012086	11:35:21.612086	11:43:17.464142	11:28:00.027403	11:43:17.264142	11:31:01.406259	11:35:15.012086
	11:00:02.604410	11:35:23.912086	11:45:48.792372	11:54:24.956129	11:29:34.337862	11:48:21.462110	11:40:54.751150	11:50:55.363178
	11:37:02.835707	11:40:45.451150			11:37:02.985707	11:50:49.763178		

Table XII. Detection Capability Using Cpms With Extracted Features For Vm8.

Attack periods			*09:37 - 9:41*	*09:42 - 9:45*	*10:06 - 10:37*	*10:43 - 10:48*	*10:48 - 11:07*
Parameters	PCA Features	CPMs	Scanning, Dictionary attack	Scanning	Transfer a file, Download a Program, Scanning, Dictionary attack	Ping the same machine, Download a Program	Failed DoS attack, Ping VM4
offset=0.05, st=2000,arl=500	F0	Mann-Whitney	No	Yes	Yes	Yes	Yes
		Cramer-von-Mises	No	Yes	Yes	Yes	Yes
	F1	Mann-Whitney	Yes	No	Yes	No	No
		Cramer-von-Mises	Yes	No	Yes	Yes	No
	F2	Mann-Whitney	~Yes (+1min)	Yes	Yes	Yes	Yes
		Cramer-von-Mises	~Yes (+1min)	Yes	Yes	Yes	Yes
	F3	Mann-Whitney	No	No	Yes	Yes	Yes
		Cramer-von-Mises	No	No	Yes	Yes	Yes



(a) Network traffic with the estimated change points locations using feature 0

(b) Network traffic with the estimated change points locations using feature 1

(c) Network traffic with the estimated change points locations using feature 2

(d) Network traffic with the estimated change points locations using feature 3

Fig.4. The plots shows the network traffic for VM8 with the estimated change points locations as a blue dashed lines and the detected change points as a red lines. These change points found using Mann-Whitney CPM

The presented result in Table XII shows that with this setting, our extracted features contributed to the CPMs' change point detection. Each feature can represent the changes in network traffic for at least two attack periods, with estimated time averaging about 1 minute from the exact attack time. Moreover, we observed that f2 performs very well and helps CPMs to detect the changes in this setting, and can present the changes in the network behavior during these attack periods.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The second conducted experiment is with offset 0.1, and the optimum set- ting found regarding the startup is similar to before, st=2000, and for ARL0 is arl=500. The detected multiple change points for both CPMs are presented in Table XII.

Table XIV illustrates the change points detection results using this setting. Although each feature contributed to detecting some period of the attack, they all failed to present the changes that happened because of the scanning attack. Therefore, this setting is recommended to use to detect different types of attacks, but not the scanning one.

Table XIII. Multiple Change Points Detected Using Change Point Methods (Cpm) For Vm8 With Riemann Rolling Scheme (Offset=0.1 Sec, Startup Delay=2000 Ms, Arl0=500 Samples

PCA Features	f_0		f_1		f_2		f_3	
IP 172.16.1.24	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time
Mann-Whitney	09:54:40.625467	10:29:23.682029	09:54:40.825467	10:29:23.682029	09:42:35.675312	10:29:23.682029	10:26:25.322594	10:29:23.682029
	10:46:59.962074	11:04:01.799066	10:49:53.297499	11:04:01.999066	09:50:08.237962	10:44:02.415555	10:43:59.515555	11:33:47.325457
	11:37:03.035707	11:50:51.513178	11:11:52.521524	11:50:57.313178	10:34:22.947925	10:57:04.777313	11:35:26.762086	11:48:21.062110
					10:46:59.762074	11:43:17.514142		
					11:14:38.870548	11:50:51.313178		
Cramer-von-Mises	09:51:43.977161	10:29:23.682029	09:51:44.077161	10:29:23.682029	09:42:35.675312	10:29:23.682029	10:26:25.322594	10:29:23.682029
	10:46:59.962074	10:59:57.704410	10:49:53.297499	10:59:57.804410	09:50:08.037962	10:44:02.415555	10:43:59.515555	11:33:47.325457
	11:37:03.035707	11:50:51.513178	11:37:03.435707	11:50:57.313178	10:34:22.947925	10:57:04.577313	11:35:26.762086	11:48:21.062110
					11:40:53.351150	11:43:17.514142		

The third experiments that we conducted is with offset 0.5, and we found the optimum setting for the startup is st=200, and arl=50000 for the ARL0. The detection time for the change points is presented in Table XV.

In this setting, the features performed better than the previous one, with at least two detections for each feature. The results of this setting are summarized in Table XVI.

The last experiments we conducted using CPMs were by choosing a setting close to the setting that we chose for Riemann chunking. We want to investigate and compare the results with small differences in the rolling window. Therefore, we chose the offset to be 1.5, and the startup is similar to before, st=200.

Table XIV. Detection Capability Using Cpms With Extracted Features For Vm8

Attack periods			*09:37 - 9:41*	*09:42 - 9:45*	*10:06 - 10:37*	*10:43 - 10:48*	*10:48 - 11:07*
Parameters	PCA Features	CPMs	Scanning, Dictionary attack	Scanning	Transfer a file, Download a Program, Scanning, Dictionary attack	Ping the same machine, Download a Program	Failed DoS attack, Ping VM4
offset=0.1, st=2000, arl=500	F0	Mann-Whitney	No	No	No	Yes	No
		Cramer-von-Mises	No	No	No	Yes	No
	F1	Mann-Whitney	No	No	No	No	Yes
		Cramer-von-Mises	No	No	No	No	Yes
	F2	Mann-Whitney	~Yes(+1min)	No	Yes	Yes	No
		Cramer-von-Mises	~Yes(+1min)	No	Yes	No	No
	F3	Mann-Whitney	No	No	Yes	Yes	No
		Cramer-von-Mises	No	No	No	No	No

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table XV. Multiple Change Points Detected Using Change Point Methods (Cpm) For Vm8 With Riemann Rolling Scheme (Offset=0.5 Sec, Startup Delay=2000 Ms, Arl0=50000 Samples

PCA Features	f_0		f_1		f_2		f_3	
IP 172.16.1.24	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time
Mann-Whitney	09:38:27.991698	09:50:07.137962	09:38:31.491698	09:50:07.137962	09:42:35.275312	09:50:07.137962	09:32:20.287716	09:50:07.137962
	09:51:43.277161	10:13:51.718338	09:51:43.777161	10:13:55.218338	09:50:08.137962	10:19:03.912579	09:50:07.137962	10:06:47.286575
	10:35:28.897294	10:36:47.712882	10:26:24.922594	10:29:28.282029	10:34:22.847925	10:36:47.212882	10:26:24.922594	10:29:14.782029
	10:46:59.962074	11:14:36.970548	10:42:23.554795	10:49:52.097499	10:46:59.462074	11:11:53.621524	10:42:27.554795	10:49:52.097499
	10:49:57.597499	11:29:30.937862	11:25:08.295442	11:26:32.854973	11:14:38.970548	11:29:30.437862	10:51:20.741746	11:26:33.354973
	10:57:03.377313	11:32:29.038038	11:35:21.362086	11:45:51.942372	11:26:36.354973	11:43:14.414142	11:31:00.956259	11:32:31.538038
	11:37:02.135707	11:39:22.672651	11:45:48.442372	11:53:22.032840			11:40:54.751150	11:50:53.013178
Cramer-von-Mises	09:38:27.991698	09:50:07.137962	09:38:31.491698	09:50:07.137962	09:42:35.275312	09:50:07.137962	09:32:20.287716	09:50:07.137962
	09:51:43.277161	10:13:51.718338	09:51:43.777161	10:13:55.218338	09:50:07.637962	10:19:03.912579	09:50:07.137962	10:06:47.286575
	10:26:24.922594	10:31:43.806592	10:26:24.922594	10:29:24.782029	10:35:28.897294	10:36:47.212882	10:26:24.922594	10:29:14.782029
	10:38:06.123231	10:49:52.097499	10:38:05.623231	10:49:52.097499	10:46:59.462074	11:14:36.970548	10:42:27.55479	10:49:52.09749
	10:46:59.962074	11:20:55.218977	10:46:59.462074	11:20:54.718977	11:14:38.970548	11:29:30.437862	10:51:20.741746	11:26:33.354973
	10:49:57.597499	11:29:30.937862	11:14:38.970548	11:29:30.437862	11:26:36.354973	11:43:14.414142	11:31:00.956259	11:32:31.538038
	10:57:03.377313	11:32:29.038038	11:35:21.362086	11:43:14.414142			11:40:54.751150	11:50:53.013178
	11:37:02.635707	11:39:22.672651	11:45:48.442372	11:53:22.032840				

Table XVI. Detection Capability Using Cpms With Extracted Features For Vm8 and for ARL0 is arl=500.

Attack periods			*09:37 - 9:41*	*09:42 - 9:45*	*10:06 - 10:37*	*10:43 - 10:48*	*10:48 - 11:07*
Parameters	PCA Features	CPMs	Scanning, Dictionary attack	Scanning	Transfer a file, Download a Program, Scanning, Dictionary attack(VM5)	Ping the same machine, Download a Program	Failed DoS attack, Ping VM4
offset=0.5, st=200,arl=50000	F0	Mann-Whitney	Yes	No	Yes	Yes	Yes
		Cramer-von-Mises	Yes	No	Yes	Yes	Yes
	F1	Mann-Whitney	Yes	No	Yes	No	No
		Cramer-von-Mises	Yes	No	Yes	Yes	No
	F2	Mann-Whitney	~Yes(+1min)	Yes	Yes	Yes	No
		Cramer-von-Mises	~Yes(+1min)	Yes	Yes	Yes	No
	F3	Mann-Whitney	No	No	Yes	Yes	Yes
		Cramer-von-Mises	No	No	Yes	Yes	Yes

The detected multiple change points results for both CPMs are presented and summarized in Table XVII. Furthermore, Table XVIII shows the detection results for each feature during the attack rounds. Using this setting, each feature can represent the changes in network traffic for at least two attack periods. However, this setting is not performing well compared to the setting used with the Riemann chunking. For instance, comparing the change points detected using f1 and f3 in this setting with the same feature in the Riemann chunking setting, we observed that both CPMs in the Riemann chunking were able to detect the changes for 4 attack periods and performed the best among other features. On the contrary, in the Riemann rolling setting, they only detected two changes, which turned out to be the lowest.

After we conducted all these experiments, we concluded that these different setting helps the 4 PCA extracted features to detect multiple change points in the network behavior. Observing these settings and understanding their impacts on the detection results is very important. Selecting a small value of the ARL0 will detect the change point faster, but with expected higher false positives. On the other hand, choosing a startup to be large will cause a delay in the detection with a high chance of false negatives.

Since cloud computing is changeable by its nature, clients' resources could be scaled up or down according to their needs. Therefore, monitoring VMs that need high network bandwidth is not similar to those that use less bandwidth. The provided

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table XVII. Multiple Change Points Detected Using Change Point Methods (Cpm) For Vm8 With Riemann Rolling Scheme (Offset=1.5 Sec, Startup Delay=200 Ms, Arlo=5000 Samples

PCA Features	f_0		f_1		f_2		f_3	
IP 172.16.1.24	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time	Change point estimated time	Detection time
Mann-Whitney	09:32:25.787716	09:49:56.637962	09:23:24.731569	09:29:11.957670	09:32:21.287716	09:32:28.787716	09:22:00.837621	09:32:06.287716
	09:49:53.637962	09:50:05.637962	09:29:11.957670	09:32:19.787716	09:42:35.275312	09:46:56.572000	09:54:38.025467	10:13:54.218338
	09:50:05.637962	09:54:39.525467	09:32:21.287716	09:35:21.888633	09:50:07.137962	09:54:39.525467	10:26:24.422594	10:29:13.782029
	09:51:42.777161	10:13:51.218338	09:49:55.137962	09:51:41.277161	10:34:21.847925	10:36:47.212882	10:38:01.123231	10:44:01.315555
	10:13:55.718338	10:19:05.912579	09:51:41.277161	09:59:08.927766	10:46:58.962074	10:52:36.573784	10:42:28.054795	10:49:50.597499
	10:20:29.058037	10:27:57.178424	10:13:55.718338	10:19:04.412579	11:14:38.470548	11:17:39.360085	10:51:19.741746	11:14:39.970548
	10:26:24.422594	10:31:42.806592	10:20:29.058037	10:27:55.678424	11:27:59.077403	11:29:31.937862	11:35:26.362086	11:40:46.251150
	10:45:29.579150	10:49:52.097499	10:42:23.554795	10:44:05.815555	11:29:36.437862	11:35:23.362086	11:40:53.751150	11:45:52.442372
	11:37:01.135707	11:40:44.751150	10:49:52.097499	11:00:00.204410	11:33:48.725457	11:43:12.914142	11:50:55.013178	11:54:24.156129
			11:25:07.795442	11:35:15.862086				
			11:26:34.854973	11:50:55.013178				
Cramer-von-Mises	09:22:00.837621	09:29:11.957670	09:32:25.787716	09:42:33.775312	09:32:21.287716	09:32:28.787716	09:22:00.837621	09:32:06.287716
	09:30:42.033764	09:32:16.787716	09:51:42.777161	09:53:13.653622	09:42:35.275312	09:46:56.572000	09:54:38.025467	10:13:54.218338
	09:32:21.287716	09:36:55.713451	10:13:55.718338	10:19:04.412579	09:51:44.277161	09:54:39.525467	10:17:45.371876	10:20:29.058037
	09:45:37.409254	09:50:04.137962	10:20:29.058037	10:27:57.178424	10:34:21.847925	10:36:47.212882	10:26:24.422594	10:29:19.782029
	09:51:42.777161	09:53:13.653622	10:42:23.554795	10:44:04.315555	10:46:58.962074	10:51:19.741746	10:38:01.123231	10:43:59.815555
	10:14:58.787659	10:19:05.912579	10:49:52.097499	10:52:36.573784	11:14:38.470548	11:17:39.360085	10:45:29.579150	10:49:52.097499
	10:20:29.058037	10:27:57.178424	11:14:38.470548	11:27:59.077403	11:27:59.077403	11:29:31.937862	11:35:26.362086	11:40:47.751150
	10:26:24.422594	10:31:44.306592	11:35:20.362086	11:43:12.914142	11:29:36.437862	11:35:23.362086	11:40:53.751150	11:44:21.053519
	10:46:58.962074	10:49:52.097499	11:45:47.942372	11:50:47.513178	11:33:48.725457	11:43:14.414142	11:50:55.013178	11:54:24.156129
	11:37:02.635707	11:40:44.751150						

Table XVIII. Detection Capability Using Cpms With Extracted Features For Vm8

Attack periods			*09:37 - 9:41*	*09:42 - 9:45*	*10:06 - 10:37*	*10:43 - 10:48*	*10:48 - 11:07*
Parameters	PCA Features	CPMs	Scanning, Dictionary attack	Scanning	Transfer a file, Download a Program, Scanning, Dictionary attack	Ping the same machine, Download a Program	Failed DoS attack, Ping VM4
offset=1.5, st=200, arl=500	F0	Mann-Whitney	No	No	Yes	Yes	No
		Cramer-von-Mises	No	Yes	Yes	Yes	No
	F1	Mann-Whitney	No	No	Yes	No	Yes
		Cramer-von-Mises	No	No	Yes	No	Yes
	F2	Mann-Whitney	~Yes(+1min)	Yes	Yes	Yes	No
		Cramer-von-Mises	~Yes(+1min)	Yes	Yes	Yes	No
	F3	Mann-Whitney	No	No	Yes	No	Yes
		Cramer-von-Mises	No	No	Yes	No	Yes

Riemann rolling scheme with these different settings provides a customizable intrusion detection system according to the VMs' operations. As expected, a lower value of the ARLo results in faster change detection, at the cost of higher false positives. Again, in practice, the user should typically decide what sort of false positive rate is acceptable and choose the ARLo appropriately.

D. Performance Analysis

To compare the effectiveness of the CloudIDS framework, we compared the performance of our suggested feature set and the Riemann-based sliding window algorithms (chunking and rolling) over different classes of attack patterns contained in the ISOT-CID dataset. In particular, we considered the initial round of attacks, involving a series of different adversarial activities launched by an external attacker and subsequently aggravated through a hijacked virtual machine (VM8).

Attack Overview and Timeline

The attack activity started with an external network scan at around 09:37, and then a successful dictionary attack on VM8 at 09:41, giving unauthorized access. Using VM8 as a leverage, the attacker performed internal reconnaissance at 09:45, performed repeated logins and file activities between 10:07–10:31, and performed additional attacks like port scans (10:30), another dictionary attack (10:36), ping sweeps, file download (10:46–10:47), and a final attempt at DoS at 11:01.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Detection Framework Summary

We employed Change Point Models (CPMs) of the R package for both single and multiple change point identification. The performance of the leading CPM algorithms, Mann-Whitney, Cramer-von-Mises, and Kolmogorov-Smirnov, is presented in the following table along with our PCA-derived features on these events.

Observations

- **Feature Contributions:** In all detection configurations, features f1 and f2 derived from PCA exhibited the greatest sensitivity in detecting behavioral anomalies, especially under dictionary and scanning attacks. Feature f0 was particularly sensitive to DoS attempts.
- **CPM Algorithm Comparison:** Mann-Whitney and Cramer-von-Mises tests outperformed other variants of CPM in early detection and reliability for both chunking and rolling modes. Kolmogorov-Smirnov was particularly strong in early-stage scanning detection.
- **Offset and Parameter Adjustment:** Smaller offsets (e.g., 0.05) and lower ARL0 values in rolling window configurations resulted in faster detections at the expense of higher false positives. Large ARL0 values yielded better stability at the expense of detection delays. These are trade-offs to be tuned according to the VM's working profile and security needs.

Ablation Study

For further understanding of the individual effects of major components in our CloudIDS model, we performed a series of ablation experiments. These experiments were conducted in order to isolate and examine the effect of PCA-derived features, CPM types, and configurations of sliding windows (chunking versus rolling) on the system's overall performance. Results confirm our design decisions and shed light on how to optimize intrusion detection for cloud environments.

E. Impact of PCA-Based Feature Extraction

We first evaluated the contribution of PCA-based dimensionality reduction to enhancing the interpretability and detection efficiency of the system. The original dataset comprises many redundant or irrelevant features, which render real-time detection less efficient. Eliminating PCA and employing raw features with CPMs resulted in dramatically higher false positives and increased detection latency, particularly in noisy network settings. Conversely, the consistent use of the top four principal components (f0–f3) always enhanced detection accuracy and lowered the average detection delay in all attack scenarios. Interestingly, features f1 and f2 performed best in identifying behavioral anomalies during dictionary attacks and internal reconnaissance.

F. Contribution of Different CPM Types

We evaluated three CPM types, Mann-Whitney, Cramer-von-Mises, and Kolmogorov-Smirnov—in single and multiple change point detection configurations. The results of ablation show that the Mann-Whitney test produced the most accurate and early detections over a broad spectrum of attack types, particularly for dictionary and port scanning attacks. Cramer-von-Mises proved more sensitive to slow-evolving behavior (e.g., internal reconnaissance and ping sweeps), and Kolmogorov-Smirnov proved to be best in detecting sudden scanning patterns in the initial stages of the attacks. Removing any of these CPMs resulted in lower detection diversity, particularly in coping with diverse and dynamic attack vectors.

G. Sliding Window Strategy: Chunking vs. Rolling

To analyze the impact of the windowing strategy, we removed either the chunking or rolling scheme and measured the detection performance. Chunking using Riemann geometry provided consistent

detection with reduced false positive rates and performed well to identify long-term changes in traffic patterns. Rolling window schemes gave more rapid, more detailed detection, particularly in cases of high frequency and subtle change. But rolling needed careful tuning of the offset and ARL0 parameters to achieve a trade-off between sensitivity and precision. The ablation established that integrating both schemes improves detection robustness across attack type and timing.

H. Sensitivity to ARL0 and Startup Parametersing

One of the crucial components of our ablation study was changing the ARL0 and startup parameters under the CPM framework. We noted that low ARL0 values accelerated detection at the expense of generating spurious alerts, especially in VMs with high variance in usage patterns. Conversely, higher ARL0 values improved stability but came with delays in responding to real threats. The startup parameter also impacted the training window of the CPMs. The disabling of these tuning mechanisms resulted in either early or missed detections, supporting the necessity of dynamic parameterization according to VM-specific network behaviors

Conclusion And Future Work

This paper introduced CloudIDS, an adaptive intrusion detection system for dynamic cloud environments. CloudIDS uses Principal Component Analysis (PCA) to extract key network traffic features and employs Change Point Models (CPMs)—specifically Mann-Whitney and Cramer-von-Mises statistics—to detect sudden behavioral shifts indicative of cyberattacks. Two Riemannian-based sliding window schemes, chunking and rolling, capture both stable and transient patterns in virtual machine (VM) network activity. Experiments with the ISOT-CID dataset show that CloudIDS effectively detects attacks such as scanning, dictionary attacks, reconnaissance, and denial-of-service with high sensitivity and low latency. Tuning parameters like Average Run Length (ARL0) and startup length significantly impacts detection performance, balancing speed and false positives. An ablation study confirmed the importance of PCA-based features and Riemannian windowing, with Mann-Whitney and Cramer-von-Mises tests outperforming other CPM variants. Rolling windows with smaller offsets enhanced detection speed but increased false positives, offering practical deployment insights. Future work will integrate CloudIDS with real-time monitoring platforms like the ELK Stack and Prometheus, and explore reinforcement learning for adaptive parameter tuning. We will also expand features with contextual metadata, investigate inter-VM correlation for distributed attack detection, and apply model optimization techniques for lightweight edge deployments. Broader benchmarking on datasets like CICIDS2017 and UNSW-NB15 will further validate CloudIDS's generalizability

References

[1] A. Sunyaev and A. Sunyaev, "Cloud computing," *Internet computing: Principles of distributed systems and emerging internet-based technologies*, pp. 195–236, 2020.

[2] R. Al Nafea and M. A. Almaiah, "Cyber security threats in cloud: Literature review," in *2021 international conference on information technology (ICIT)*, IEEE, 2021, pp. 779–786.

[3] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Applied Soft Computing*, vol. 92, p. 106301, 2020.

[4] A. Goswami, R. Patel, C. Mavani, and H. K. Mistry, "Intrusion Detection and Prevention for Cloud Security," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 12, no. 2, pp. 556–563, 2024.

[5] A. N. Jaber, S. Anwar, N. Z. Bin Khidzir, and M. Anbar, "The importance of ids and ips in cloud computing environment: Intensive review and future directions," in *Advances in Cyber Security: Second International Conference, ACeS 2020, Penang, Malaysia, December 8-9, 2020, Revised Selected Papers 2*, Springer, 2021, pp. 479–491.

[6] M. A. Hatef, V. Shaker, M. R. Jabbarpour, J. Jung, and H. Zarrabi, "HIDCC: A hybrid intrusion detection approach in cloud computing," *Concurrency and Computation: Practice and*

Experience, vol. 30, no. 3, p. e4171, 2018.

[7] H. Satilmiş, S. Akleylek, and Z. Y. Tok, "A Systematic Literature Review on Host-Based Intrusion Detection Systems," *Ieee Access*, vol. 12, pp. 27237–27266, 2024.

[8] A. Thakkar and R. Lohiya, "Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system," *Information Fusion*, vol. 90, pp. 353–363, 2023.

[9] N. S. K. Anumukonda, R. K. Yadav, and N. S. Raghava, "Hypervisor Based Intrusion Detection Using Enhanced Radial Basis Neural Network on Cloud Environment," in *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, IEEE, 2024, pp. 1–6.

[10] C. Saadi and H. Chaoui, "A new approach to mitigate security threats in cloud environment," in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, 2017, pp. 1–7.

[11] B. Alouffi, M. Hasnain, A. Alharbi, W. Alosaimi, H. Alyami, and M. Ayaz, "A systematic literature review on cloud computing security: threats and mitigation strategies," *Ieee Access*, vol. 9, pp. 57792–57807, 2021.

[12] W. Li, S. Tug, W. Meng, and Y. Wang, "Designing collaborative blockchained signature-based intrusion detection in IoT environments," *Future Generation Computer Systems*, vol. 96, pp. 481–489, 2019.

[13] T. Zoppi, A. Ceccarelli, L. Salani, and A. Bondavalli, "On the educated selection of unsupervised algorithms via attacks and anomaly classes," *Journal of Information Security and Applications*, vol. 52, p. 102474, 2020.

[14] S. Sonawane, "Rule based learning intrusion detection system using KDD and NSL KDD dataset," *Prestige International Journal of Management & IT-Sanchayan*, vol. 4, no. 2, pp. 134–144, 2015.

[15] Z. Chiba, N. Abghour, K. Moussaid, and M. Rida, "Intelligent approach to build a Deep Neural Network based IDS for cloud environment using combination of machine learning algorithms," *computers & security*, vol. 86, pp. 291–317, 2019.

[16] R. Thomas and D. Pavithran, "A survey of intrusion detection models based on NSL-KDD data set," *2018 Fifth HCT Information Technology Trends (ITT)*, pp. 286–291, 2018.

[17] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Communication Networks*, vol. 2018, no. 1, p. 4943509, 2018.

[18] A. Guezzaz, S. Benkirane, M. Azrour, and S. Khurram, "A reliable network intrusion detection approach using decision tree with enhanced data quality," *Security and Communication Networks*, vol. 2021, no. 1, p. 1230593, 2021.

[19] T. Arvind, "A survey on building an effective intrusion detection system (IDS) using machine learning techniques, challenges and datasets," *International Journal for Research in Applied Science and Engineering Technology*, pp. 1473–1478, 2020.

[20] A. Devarakonda, N. Sharma, P. Saha, and S. Ramya, "Network intrusion detection: A comparative study of four classifiers using the NSL-KDD and KDD'99 datasets," in *Journal of Physics: Conference Series*, IOP Publishing, 2022, p. 12043.

[21] K. M. Al-Gethami, M. T. Al-Akhras, and M. Alawairdhi, "Empirical evaluation of noise influence on supervised machine learning algorithms using intrusion detection datasets," *Security and Communication Networks*, vol. 2021, no. 1, p. 8836057, 2021.

[22] C. Liu, Z. Gu, and J. Wang, "A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning," *Ieee Access*, vol. 9, pp. 75729–75740, 2021.

[23] S. Sharma, Y. Gigras, R. Chhikara, and A. Dhull, "Analysis of NSL KDD dataset using classification algorithms for intrusion detection system," *Recent Patents on Engineering*, vol. 13, no. 2, pp. 142–147, 2019.

[24] N. Garcia, T. Alcaniz, A. González-Vidal, J. B. Bernabe, D. Rivera, and A. Skarmeta, "Distributed

real-time SlowDoS attacks detection over encrypted traffic using Artificial Intelligence," *Journal of Network and Computer Applications*, vol. 173, p. 102871, 2021.

[25] H. Hourani and M. Abdallah, "Cloud computing: legal and security issues," in *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, IEEE, 2018, pp. 13–16.

[26] F. Palumbo, G. Aceto, A. Botta, D. Ciuonzo, V. Persico, and A. Pescapé, "Characterizing Cloud-to-user Latency as perceived by AWS and Azure Users spread over the Globe," in *2019 IEEE global communications conference (GLOBECOM)*, IEEE, 2019, pp. 1–6.

[27] N. H. Hussein and A. Khalid, "A survey of cloud computing security challenges and solutions," *International Journal of Computer Science and Information Security*, vol. 14, no. 1, p. 52, 2016.

[28] J. Martínez Torres, C. Iglesias Comesaña, and P. J. García-Nieto, "Machine learning techniques applied to cybersecurity," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2823–2836, 2019.

[29] M. Fouda, R. Ksantini, and W. Elmedany, "A novel intrusion detection system for internet of healthcare things based on deep subclasses dispersion information," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8395–8407, 2022.

[30] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine learning and deep learning approaches for cybersecurity: A review," *IEEE Access*, vol. 10, pp. 19572–19585, 2022.

[31] A. A. Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain, "Intrusion detection system for healthcare systems using medical and network data: A comparison study," *IEEE Access*, vol. 8, pp. 106576–106584, 2020.

[32] I. Prokopenko, "Nonparametric change point detection algorithms in the monitoring data," in *Advances in Computer Science for Engineering and Education IV*, Springer, 2021, pp. 347–360.

[33] S. Deldari, D. V Smith, H. Xue, and F. D. Salim, "Time series change point detection with self-supervised contrastive predictive coding," in *Proceedings of the web conference 2021*, 2021, pp. 3124–3135.

[34] B. L. Lavy, R. C. Weaver, and R. R. Hagelman III, "Using the change point model (CPM) framework to identify windows for water resource management action in the lower Colorado River basin of Texas, USA," *Water*, vol. 14, no. 1, p. 18, 2021.

[35] T. Budhraja, B. Goyal, A. Kilaru, and V. Sikarwar, "Fuzzy clustering-based efficient classification model for large TCP dump dataset using hadoop framework," in *Proceedings of International Conference on ICT for Sustainable Development: ICT4SD 2015 Volume 1*, Springer, 2016, pp. 427–437.

[36] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," *arXiv preprint arXiv:1702.05538*, 2017.

[37] I. Cano, S. Aiyar, and A. Krishnamurthy, "Characterizing private clouds: A large-scale empirical analysis of enterprise clusters," in *Proceedings of the Seventh ACM Symposium on Cloud Computing*, 2016, pp. 29–41.

[38] M. Kiperberg, R. Leon, A. Resh, A. Algawi, and N. J. Zaidenberg, "Hypervisor-based protection of code," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2203–2216, 2019.

[39] S. Yu, X. Li, Y. Feng, X. Zhang, and S. Chen, "An instance-oriented performance measure for classification," *Information Sciences*, vol. 580, pp. 598–619, 2021.

[40] W. Duch, T. Wieczorek, J. Biesiada, and M. Blachnik, "Comparison of feature ranking methods based on information entropy," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, IEEE, 2004, pp. 1415–1419.

[41] X. Tong, C. Kang, and Q. Xia, "Smart metering load data compression based on load feature identification," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2414–2422, 2016.

[42] D. Machiwal, S. Kumar, H. M. Meena, P. Santra, R. K. Singh, and D. V. Singh, "Clustering of

Research Article

rainfall stations and distinguishing influential factors using PCA and HCA techniques over the western dry region of India," *Meteorological Applications*, vol. 26, no. 2, pp. 300–311, 2019, doi: 10.1002/met.1763.

[43] F. Salo, A. B. Nassif, and A. Essex, "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection," *Computer Networks*, vol. 148, pp. 164–175, 2019, doi: 10.1016/j.comnet.2018.11.010.

[44] M. Thalmann, A. S. Souza, and K. Oberauer, "How does chunking help working memory?," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 45, no. 1, p. 37, 2019.

[45] S. B. Chae, *Lebesgue integration*. Springer Science & Business Media, 2012.

[46] A. Aldribi, I. Traore, P. G. Quinan, and O. Nwamuo, "Documentation for the isot cloud intrusion detection benchmark dataset (isot-cid)," *University of Victoria*, 2020.

[47] A. Aldribi, I. Traore, and B. Moa, "Data sources and datasets for cloud intrusion detection modeling and evaluation," *Cloud computing for optimization: foundations, applications, and challenges*, pp. 333–366, 2018.

[48] Altowaijri, S.M. and El Touati, Y. 2024. Securing Cloud Computing Services with an Intelligent Preventive Approach. Engineering, Technology & Applied Science Research. 14, 3 (Jun. 2024), 13998–14005. DOI:https://doi.org/10.48084/etasr.7268.

[49] Al-mugern, R., Othman, S.H., Al-Dhaqm, A. and Ali, A. 2024. A Cloud Forensics Framework to Identify, Gather, and Analyze Cloud Computing Incidents. *Engineering, Technology & Applied Science Research*. 14, 3 (Jun. 2024), 14483–14491. DOI:https://doi.org/10.48084/etasr.7185